WL-TR-91-1038

EVALUATION AND VALIDATION GUIDEBOOK
Version 3.0

Peter G. Clark
Bard S. Crawford

TASC
55 Walker's Brook Drive
Reading MA 01867

May 1991

Interim Report

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Raymond Szymanski
RAYMOND SZYMANSKI
Project Engineer

25 March 1991
Date

FOR THE COMMANDER

Charles H. Klinger

Date

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/AAAF , WPAFB, OH 45433-6543 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1991	Interim

Evaluation & Validation Guidebook
Version 3.0

5. FUNDING NUMBERS
C-F33615-85-C-1812
PE-63756D
PR-2853
TA-01
WU-01

Peter C. Clark
Bard S. Crawford

TASC
55 Walker's Brook Drive
Reading MA 01867

8. PERFORMING ORGANIZATION REPORT NUMBER
TASC
TR 5234-4

Raymond Szymanski
WL/AAAF-3 (513) 255-3947
WPAFB OH 45433-6543

9. SPONSORING/MONITORING AGENCY REPORT NUMBER
WL-TR-91-1038

This report supercedes the Evaluation & Validation Guidebook
Version 2.0

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution is unlimited

The purpose of the E&V Guidebook is to provide information that will help users to assess APSEs and APSE components by:

(1) Assisting in the selection of E&V procedures, the interpretation of results, and integration of analyses and results,
(2) Describing E&V procedures and techniques developed by the E&V Project, and
(3) Assisting in the location of E&V procedures and techniques developed outside the E&V Project.

All E&V procedures and techniques found in the Guidebook are referenced by the indexes contained in the companion document called the E&V Reference Manual.

Ada Programming Support Environment (APSE)
Ada
Evaluation, Validation, Metrics, ACEC

15. NUMBER OF PAGES
237

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclass	Unclass	Unclass	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to **stay within the lines to meet optical scanning requirements.**

Block 1. Agency Use Only (Leave Blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency. Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ..., To be published in .... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availablity Statement. Denote public availability or limitation. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR)

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - DOD - Leave blank
DOE - DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports
NASA - NASA - Leave blank
NTIS - NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

EXECUTIVE SUMMARY

The Ada community, including Government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and their components, and to determine their conformance to applicable standards (e.g., MIL-STD-1838A, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government-sponsored, professional society-sponsored, or private effort. The purpose of the APSE Evaluation and Validation (E&V) Project is to provide a focal point for addressing the need by:

(1) Identifying and defining specific technology requirements,

(2) Developing selected elements of this technology,

(3) Encouraging others to develop additional elements, and

(4) Collecting information describing elements which already exist.

This information will be made available to DoD components, other government agencies, industry and academia.

The purpose of the E&V Guidebook (this document) is to provide information that will help users to assess APSEs and APSE components by:

(1) Assisting in the selection of E&V procedures, the interpretation of results, and integration of analyses and results,

(2) Describing E&V procedures and techniques developed by the E&V Project, and

(3) Assisting in the location of E&V procedures and techniques developed outside the E&V Project.

All E&V procedures and techniques found in the Guidebook are referenced by the indexes contained in the companion document called the E&V Reference Manual.

Chapters 1 through 4 of this document provide a general introduction to E&V, describe the structure of the Guidebook and how to use it, discuss the issue of integrating the results of multiple assessments, and provide synopses of relevant E&V literature. Chapter 5 and later

chapters are built around a standard format and each chapter contains all the assessment procedures and techniques associated with a particular group of tools or toolsets to be assessed, such as Compilation System Assessors or Test System Assessors. The assessment procedures are described and in some instances can be applied directly from the information given in the Guidebook. In other cases, the user is directed to a primary reference for more information.

Readers should not infer approval by the E&V Team or its sponsors, because a tool or technique is included in the collection, or disapproval, because a tool or technique is not included. Nor should readers infer any judgement as to the relative importance of an entry based on its position in a chapter. Readers who know of instances of E&V technology not reported here are urged to contact the E&V Project Leader, in the manner described below.

Yearly updates and extensions to this document are planned. Therefore, comments and suggestions are welcome. Please send your comments on the Request for User Feedback at the back of the Guidebook by regular mail to Mr. Raymond Szymanski, WL/AAAF, Wright Patterson AFB, OH 45433-6543, or, preferably, the same information electronically to szymansk@ajpo.sei.cmu.edu.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 PURPOSE OF GUIDEBOOK

This document is a product of the Ada Programming Support Environment (APSE) Evaluation and Validation (E&V) Project sponsored by the Ada Joint Program Office. It is one of a pair of companion documents known as the E&V Reference System, consisting of:

- E&V Reference Manual
- E&V Guidebook.

The subject of both documents is the assessment of APSEs and their components. Specific assessment techniques typically fall into one of two categories: evaluation (assessment of performance and quality) and validation (assessment of conformance to a standard).

The purpose of the Guidebook is to provide a collection of information to support a variety of E&V users in the following ways. It should help them:

- Gain an overall understanding of APSE assessment, in particular, the selection of appropriate E&V procedures, the interpretation of test results, and the integration of analyses and results.
- Apply the various E&V procedures and techniques developed under E&V Project sponsorship.
- Find the primary sources for those E&V procedures and techniques not developed by the E&V Project or not fully explained within the Guidebook (due to space or other constraints).

The Reference Manual includes many "pointers" to sections in the Guidebook and other documents which describe E&V techniques in much the same way that a card catalog does in a library. Figure 1.1-1 illustrates the relationship between the documents.

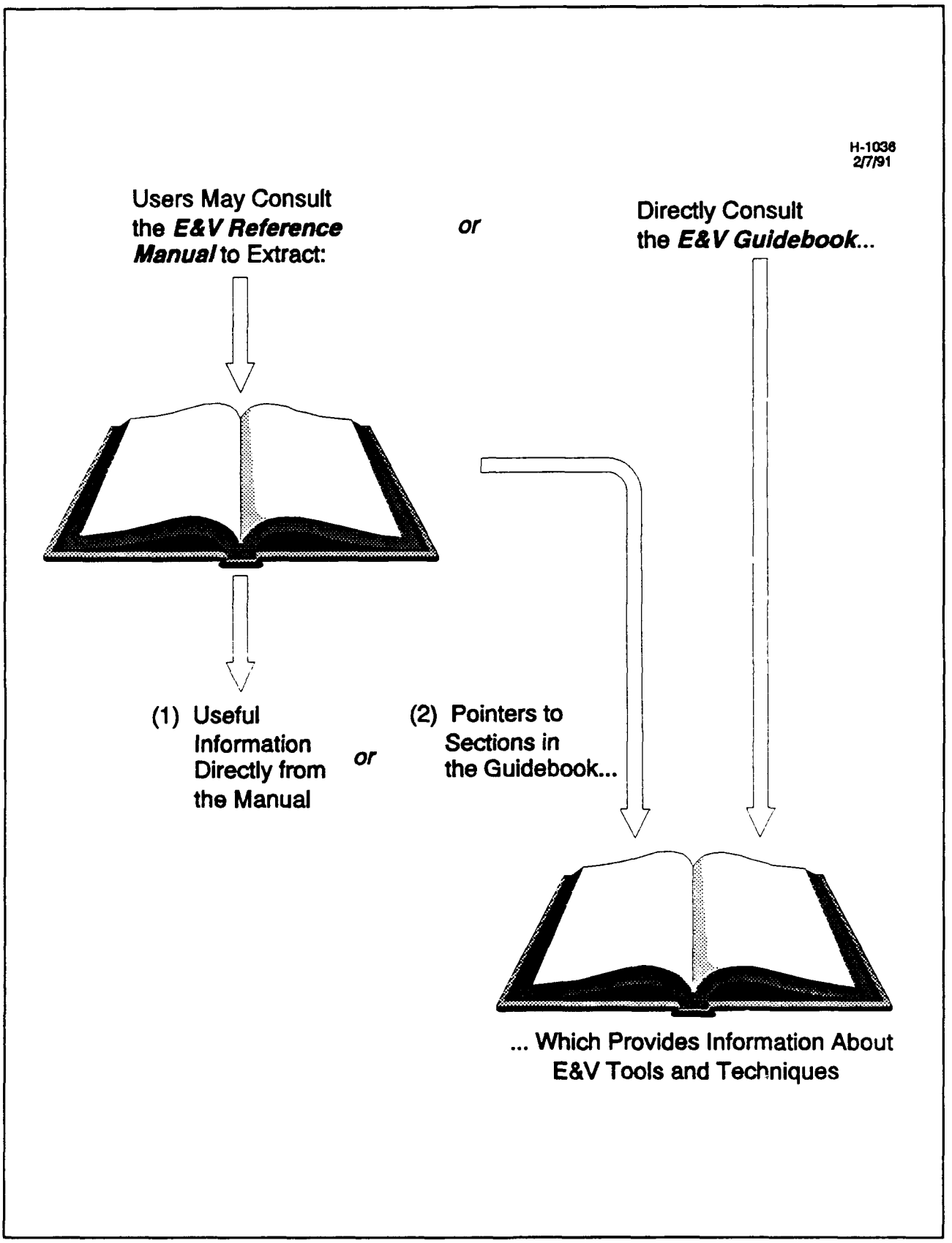

Figure 1.1-1 Relationship Between Reference Manual and Guidebook

1.2 THE NEED FOR E&V TECHNOLOGY

Technology for the assessment of APSEs and APSE components (tools) is needed because of the difficulty in assessing APSEs and because of the importance of the decisions made based on these assessments. The importance of an APSE selection is evident when one considers the large, critical, Ada-based systems to be developed in the coming years. The effectiveness, reliability, and cost of these systems will be strongly influenced by the environments used to develop and maintain them. From the point of view of a software developing organization, the decision to select an APSE can be an important investment decision with long-lasting influence on a number of projects and the organization's operating procedures, training, and competitiveness. From the point of view of a software maintenance organization, the environment used will strongly influence the organization's effectiveness, as well as the cost of its operations and training.

The difficulty of assessing APSEs and tools exists for several reasons. First, an APSE represents very complex technology with many elements, which can be assessed individually or in combination. Second there is a confusing diversity of choice with respect to individual tools, tool sets, or "whole APSEs"; and there are a number of ways of viewing APSEs; see Chapter 3 of the E&V Reference Manual [@RM 3].* Third, the state of the art of APSE architecture and of some categories of tools (e.g., graphic design tools) is undergoing rapid change. Finally, there is a lack of historical data relevant to APSEs, partly because of the general pace of technological change and partly because we are dealing with Ada, a relatively new implementation language. E&V technology provides methods and techniques to overcome these difficulties and provides a basis for determining performance and other attributes of APSEs.

In addition to the need for assessment technology itself, there is a need for information about this technology. Potential buyers and users of APSEs and tools need a framework for understanding APSEs and their assessment, as well as information about specific assessment techniques. Similarly, vendors of tools and APSEs need to be aware of the deficiencies of current products, as well as the criteria to be used in the assessment of future products. Such awareness on both sides, expressed in a common terminology, should speed up the evolution of better software development environments.

---

*The format used for references is associated with the "formal grammar" used beginning with Chapter 5. See further explanations in Appendix C.

1.3 BACKGROUND

In June 1983 the Ada Joint Program Office (AJPO) proposed the formation of the E&V Project and a tri-service E&V Team, with the Air Force designated as lead service. In October 1983 the Air Force officially accepted responsibility as lead service and designated the Air Force Wright Aeronautical Laboratories (AFWAL) at Wright Patterson Air Force Base as lead organization. In April 1984 an E&V Workshop was held at Airlie, Virginia. The purpose of the workshop was to solicit participation of industry representatives in the E&V Project. Many of the participants in the workshop have chosen to remain involved as Distinguished Reviewers, and additional industry participants have subsequently become involved in E&V Team activities.

The E&V Project publishes an annual public report. The following paragraph is quoted from the 1987 version [@Szymanski 1987] of the report:

> The Ada community, including government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and components and to determine their conformance to applicable standards (e.g., DoD-STD-1838, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government-sponsored, professional society-sponsored, or private effort. The purpose of the APSE Evaluation and Validation (E&V) Task is to provide a focal point for addressing the need by (1) identifying and defining specific technology requirements, (2) developing selected elements of the required technology, (3) encouraging others to develop some elements, and (4) collecting information describing existing elements. This information will be made available to DoD components, other government agencies, industry, and academia.

The team public reports contain much additional information for the interested reader. See for example, the "DoD APSE Analysis Report" [@Castor 1984], the "Requirements for the Evaluation and Validation of Ada Programming Support Environments, Version 2.0" [@Szymanski 1987], and the "Tools and Aids Document, Version 1.0" [@Szymanski 1987], which are synopsized in Chapter 4 [4.3, 4.5, 4.6].

Three competitive contracts have been awarded under the E&V Project. These are:

- Technical Support contract—awarded June 1985
- Ada Compiler Evaluation Capability (ACEC) contract—awarded February 1987
- CAIS Implementation Validation Capability (CIVC) contract—awarded May 1987.

The major purpose of the first of these contracts is to create and update elements of the E&V Reference System, including this document. The purpose of the second and third contracts is to create two additional elements (ACEC and CIVC) of the needed E&V technology.

1.4 ORGANIZATION OF THE GUIDEBOOK

Chapter 2 provides a general description of the structure and use of the Guidebook.

Chapter 3 provides high-level guidance to users who may need assistance in selecting instances of the technology and integrating the results of its application.

Chapter 4 provides synopses of other documents or activities that are either too broad in scope to fit within one of the later chapters or are of historical importance to E&V activities.

Chapter 5 and subsequent chapters are "formal chapters" that describe or refer to specific instances of E&V technology. Each of the formal chapters contains all of the procedures and techniques associated with a particular group of tools or toolsets to be assessed, such as Compilation System Assessors or Test System Assessors. A standard format based on a "formal grammar" is used in presenting this material. This formal grammar is explained further in Appendix C.

Appendices A, B, and C contain a list of citations, a list of acronyms and abbreviations, and a definition of the formal grammar used in the formal chapters, respectively. *Appendix D* contains the list of vendors and agents of assessment tools who are the primary sources of E&V technology.

2. STRUCTURE AND USE OF THE GUIDEBOOK

This chapter provides a brief explanation of the structure and uses of the E&V Guidebook. It is expected that many users have first consulted the E&V Reference Manual (see Fig. 1.1-1) and come to the Guidebook with a specific chapter and section number in hand, prepared to read about a specific instance of E&V technology. A user following this path does not particularly care about the overall structure of the document. Other users, however, may come to the document with a less narrowly-defined objective. An attempt has been made, with such users in mind, to make the Guidebook easy to use as a stand-alone document.

2.1 STRUCTURE

The Guidebook structure may be considered as having four major subdivisions, as follows:

- Introductory Material (Chapters 1 and 2)
- General Background Material (Chapters 3 and 4)
- Specific E&V Technology Descriptions (Chapters 5 and beyond)
- Appendices.

Chapters 1 and 2 are used to introduce the document and its structure. The general background material is used to introduce the general subject of APSE assessment. Chapter 3 is an "essay" designed to help users who are faced with the question of how to evaluate an APSE as a whole, or how to compare several APSEs with the objective of selecting one. (Chapter 3 of the E&V Reference Manual, dealing with whole APSE assessment issues, is a "companion essay" that provides complementary background material.) Chapter 4 provides a different kind of background material. It may be considered a "guide to the literature" of APSE assessment. It contains synopses of documents that fall into one of two categories. One category is that of documents that contain no specific instances of E&V technology, but contain

generally useful background material. The other category is that of documents that contain or discuss multiple instances of E&V technology, which are individually covered in multiple parts of the later, formal chapters. These multiple instances can be thought of as children of a common parent. In order to avoid the redundancy of summarizing the parent document many times, the Chapter 4 synopsis is provided as a common point to which all the children may refer. Each "synopsis text frame" in Chapter 4 has the following parts:

- Citation: (the primary reference)
- Synopsis: (brief description)
- Methods: (references to specific instances of E&V technology, if any).

The formal chapters (Chapters 5 and beyond), which comprise the main bulk of the Guidebook, describe or summarize specific instances of E&V technology. The chapter subjects and titles were chosen to be meaningful and intuitive to users of the Guidebook. Thus, they focus on the subject of assessment (e.g., Compilation System, Test System, Ada Design Support System, etc) rather than the method of assessment (e.g., formal validation, subjective evaluation, etc). Within each chapter there are, in general, multiple instances of assessment technology. Some may be examples of evaluation techniques, others may be examples of validations, others may be mixtures of the two. *Readers should not infer approval* by the E&V Team or its sponsors, because a tool or technique is included in the collection, *or disapproval*, because a tool or technique is not included. Readers who know of instances of E&V technology not reported here are urged to contact the E&V Project chairman, in the manner described in the Executive Summary. The separate instances within a chapter are simply placed there in chronological order, indicating the relative timing of the material's first appearance in the Guidebook. *Readers should not infer any judgment as to relative importance based on order.* Each chapter thus provides a dynamically growing section of the Guidebook. Old sections will not be thrown away or replaced by new sections describing newer techniques. Old sections may, however, be updated if a particular vendor or agent has updated material describing a technique, or has improved the technique itself without fundamentally changing the approach. Each "technique text frame" in the formal chapters has the following parts:

- Purpose:
- Primary References:
- Host/OS: (if applicable)
- Vendors/Agents: (if applicable)
- Method:
  Inputs:
  Process:
  Outputs:

The final major subdivision (the appendices) require little explanation here. The formal grammar described in Appendix C need not concern most users. It was employed because of the possibility of a future on-line, electronic version of the Reference System, supported by advanced updating and information retrieval techniques.

2.2 EXAMPLE USES

Instances of E&V technology may be found in two ways. A user may consult the Guidebook directly, or may first consult the E&V Reference Manual, as pictured in Figure 1.1-1. A user who comes directly to the Guidebook would typically first look at the Table of Contents. For example, a user interested in evaluating compiler performance would naturally look under Chapter 6 "Compilation System Assessors." The titles of Sections 6.2, "IDA Benchmarks," and 6.3, "Ada Compiler Evaluation Capability (ACEC)," would probably suggest themselves as relevant to this user's needs—as indeed they are.

Alternatively, the user may consult the E&V Reference Manual, which is designed to help find E&V techniques in the same way that the card catalog helps people find books in the library. For example, the Reference Manual contains both a Function Index and an Attribute Index, each of which contains cross references to elements in the other. One element of the Function Index is the function "Compilation," which is cross-referenced to a number of relevant attributes. Under the particular function-attribute pair "Compilation-Processing Effectiveness"

are listed a number of Guidebook references. Among these are the same two Sections, 6.2 and 6.3, of the Guidebook, mentioned in the previous paragraph. The user following this procedure could pick up the Guidebook and go directly to these two sections or "text frames" and find summary information concerning the IDA Benchmarks test suite and the ACEC test suite, respectively.

2.3 BIAS IN EVALUATION

Some elements of bias are inherent in all evaluation techniques. Examples of such elements are given in the following paragraph. It is important that users of evaluation techniques be aware of these built-in biases and use caution in the interpretation of results. A tool or APSE that is "different" may receive an unfair evaluation because of an unintended bias against new technology or a new concept of operations. The effects of bias can be minimized, but not eliminated, by careful design of experiments. In some situations certain elements of bias are actually desirable, as discussed in the final paragraph of this section.

Consider, for example, a whole-APSE evaluation based on a series of structured experiments involving various portions of the life cycle. The items to be evaluated are competing commercial software products—collections of tools integrated in some way. The experiments are built around a model project, partially completed, and instructions to perform specific life-cycle activities such as test and integration, configuration management, response to a change in system requirements, or documentation updates. The outcome of such experiments are inevitably influenced by factors that are not characteristics of the software products under evaluation. These factors include: the skill and experience of the evaluation team members, the management ability of the team leaders, the software development methods ordinarily favored by the team members (as opposed to those best supported by the APSEs under evaluation), the application domain of the model project (as opposed to those in which team members are experienced), and other surrounding environmental factors.

The influence of the factors listed above can be controlled to some extent. For example, it is possible to employ a sequence in which: in Phase 1 Team 1 does Model Project M on APSE A while Team 2 does Model Project N on APSE B; in Phase 2 the teams exchange roles; in Phase 3 they compare notes and write a joint evaluation report. This sort of approach can

be useful in removing bias, but is of course very expensive.

Before embarking on an APSE evaluation it is important to have a clear understanding of the purpose of the evaluation. It is unlikely, for example, that the purpose of an APSE evaluation project will be to select "the best APSE" in some general, global sense. It is more likely that the purpose will be to select the best APSE for a particular project or sequence of projects, to be used by a particular organization (with its unique history and preferences), in a particular application domain. It is also possible that there is only one APSE available or that an APSE has already been chosen. In this case the purpose of the evaluation includes obtaining a better understanding of the characteristics of the APSE and the risks and costs associated with its use in a particular application domain and with a particular development methodology. In such cases it is quite legitimate for the evaluation to reflect organizational and individual "biases." It is, after all, a particular group of individuals who will be asked to use the APSE in a productive way. If they have a choice, they will want to choose an APSE that supports their style. If the choice is already made, they will need to understand how the given APSE supports, or fails to support, their preferred methods of operation. Thus, a "biased evaluation" can be a desirable and necessary objective.

3. INTEGRATION OF APSE ASSESSMENTS

The purpose of this chapter is to provide high-level guidance for the user of the E&V Reference System (Reference Manual and Guidebook) who is interested in evaluating an APSE as a whole, or in comparing several APSEs with the objective of selecting one. While the "formal chapters" (beginning with Chapter 4 of the Reference Manual and Chapter 4 of the Guidebook) provide assistance in locating, defining, and assessing many individual aspects of APSEs, they do not provide an overall approach to weighing and combining the results of such assessments. Section 3.1 briefly discusses some relevant general background material. Section 3.2 discusses some earlier, partial efforts aimed at an integrated approach. Section 3.3 provides some additional guidance leading to a comprehensive, integrated approach.

It is necessary, first, to distinguish the subject of this chapter—integrated whole-APSE assessment—from the subject of Chapter 13—specific "Whole-APSE Assessors." The integrated form of whole-APSE assessment (Section 3.3) involves a combining or mixing together of the results of individual assessment steps to arrive at a decision. These individual steps may be oriented toward specific functions or tools, or may be oriented toward a "whole APSE," in relation to specific attributes or the APSE's performance in a specific life-cycle phase or activity. Thus, a whole APSE assessor (Chapter 13) might be used to evaluate the APSE's capability to support a project team during one major activity, such as preliminary design. The results of such an assessment would become one of the weighted factors of an integrating process leading to a major decision.

3.1 GENERAL BACKGROUND

Chapter 4 of this Guidebook contains synopses of books, articles, and documents. Some of these have historical value and are also indirectly relevant to the topic of an integrated approach to APSE evaluation because they provide definitions of an APSE or highlight issues that may be important during APSE evaluations. The Stoneman document [@Buxton 1980] defines an APSE as a layered system and includes some discussion of evaluation criteria. The Common APSE Interface Set (CAIS) definition documents [@DoD 1986, @MIL 1989] describe

proposed interface requirements for interfaces that exist between layers of an APSE. The motivation for these interface requirements is to support the transportability of tools and project databases from one APSE to another. The book "Life Cycle Support in the Ada Environment" by McDermid and Ripken [@McDermid 1984] takes a top-down approach to defining a "coherent APSE," starting with requirements for a coherent life-cycle methodology; see synopsis [4.10]. Several papers in an IEEE Tutorial [@Wasserman 1981] provide relevant observations on desirable characteristics and major issues for Ada support environments; see synopses [4.11, 4.12, 4.14]. A more recent survey paper "Characteristics and Functions of Software Engineering Environments: An Overview" [@Houghton 1987] provides a broad discussion of environments and the state of the art; see synopsis [4.15]. Chapter 3 of the E&V Reference Manual [@RM: Whole APSE Assessment Issues 3.] presents various ways of viewing an APSE and key whole-APSE attributes.

3.2 EARLY EFFORTS AT INTEGRATED APSE ASSESSMENT

The following quotation is from a paper by Henderson and Notkin [@Henderson 1987]:

> Perhaps the biggest failing of environments research and development to date is the general lack of scientific evaluation of existing environments. Evaluation approaches and actual evaluations are beginning to appear, but relatively little effort has been given to this undeniably fundamental subject.

Some early efforts are mentioned briefly below.

The Software Engineering Institute (SEI) has developed a methodology [@Weiderman 1987] to evaluate certain aspects of APSEs. The methodology centers around the execution of several experiments in the environment(s) to be evaluated. The experiments are designed in a generic fashion and must be tailored or "instantiated" for each specific environment; see synopsis [4.13]. The early applications, by the SEI, of this methodology were aimed at limited objectives (see [7.9, 8.2, 10.1, and 11.2]), and are not examples of integrated whole-APSE assessments. However, the SEI methodology has been applied by TRW to a broad-gauged, whole-APSE selection process; see [@Gray 1987]. It is apparent that industry has devoted resources internally to comparative assessment of commercial APSEs. However, other than

Gray's paper, little has yet been published in the open literature describing the techniques employed.

The book "Selecting an Ada Environment" [@Lyons 1986], written by the Ada Europe Environment Working Group, provides background discussion about a broad range of topics. In each chapter and section it provides a list of questions to be asked about the environment under consideration; see synopsis [4.9]. Some of the chapters and questions listed have a definite "integrated whole-APSE assessment" flavor. The whole-APSE checklist [14.2] has been adapted from material of this book.

3.3 TOWARDS A COMPREHENSIVE APPROACH

The published literature on assessment of software engineering environments does not include descriptions of "decision support" oriented approaches. (But, see the dissertation [@Lawlis 1989].) A decision support system is one that leads a user through a structured framework that includes weighting factors and decision criteria, and supports a final decision process. As applied to APSE assessment this kind of approach would support a final decision, such as, whether a single APSE under consideration is "good enough," or which of several APSEs under consideration is "best."

The following characteristics appear to be appropriate for a decision support system designed for integrated APSE assessment:

- The system should allow the specification of a list of "essential features" that are absolutely required for the contemplated application or family of applications. Ideally, each of these essential features would be subject to a question or test that yields an unambiguous "yes/no" result—yes, the required feature is present, or no, it is missing.

- The system should allow the specification of a second list of attributes and function-attribute pairs that represent desirable features or criteria, which should be involved in an integrated assessment.

- The system should allow for specification of "weights" to be applied to each attribute and function-attribute pair in the second list. The weights will typically be chosen subjectively by the assessment participants.

- The system should include a mechanism to document/identify the method of assessment used for every test/metric to be employed in addressing every essential and desirable feature.

- The system should include a well-defined method of combination, leading to an overall set of pre-decision results. For example, the results may be summarized in two lines as in:
  1) satisfies all essential requirements (listed in Table A)
  2) scores 72 out of possible 100 (based on weights in Table B).

The characteristics outlined above represent a general framework that can be applied very differently by different users. At one extreme is a decision maker with little time or resources, who focuses on a short list of essential features only, and accepts answers supplied by vendors or vendor documentation. At the other extreme is a team of APSE assessors who conduct a comprehensive, detailed set of tests and "model project" experiments and expend multiple person-years of effort in a comparative, hands-on assessment of competing APSEs.

It is also possible that two assessment teams applying equal resources might differ greatly in the manner of their assessments. One might view the APSE as a support system for a particular life-cycle methodology adopted by its organization. Another might view the APSE as a project database management system. These two teams would be likely to use very different tests, or very different weights where the same tests are used. Neither is necessarily right or wrong. In the final analysis, it is the software developer's responsibility to understand his own application area and the most critical attributes of his development support environment.

4. SYNOPSES

The purpose of this chapter is to provide a single place in the Guidebook for synopses of documents (or other resources), which have too broad a scope to fit within one of the subsequent Chapters. In some cases the subject document appears only in this Chapter because it does *not* contain specific instances of E&V technology. For example, the Stoneman document [@Buxton 1980] does not deal with evaluation or validation of APSEs, but it has general historical importance to the entire field of Ada environments and has been selected as the first document to be synopsized. Synopses describing APSE standards that may require validation are also included, such as the entries for the CAIS and IRDS. Finally, a particular document may contain *multiple* instances of E&V technology, which are themselves summarized or referenced in multiple parts of the Guidebook. These multiple instances can be thought of as children of a common parent. In order to avoid the redundancy of summarizing the parent document many times, the Chapter 4 synopsis is provided as a common point to which all the children may refer. The formal grammar used to structure the entries in subsequent chapters includes, therefore, a mechanism for referring back to the synopsis contained in Chapter 4. Similarly, after each synopsis there is a provision for forward references to specific techniques (if any) described in later chapters.

Most of the documents synopsized in this chapter are readily available through public sources. A few of them may be difficult or impossible to obtain for some readers; these were included because the synopsis itself was judged to be helpful in filling in a piece of the historical background.

4.1 STONEMAN

Citations: [Buxton 1980] J.N. Buxton, "Requirements for Ada Programming Support Environments—STONEMAN," U.S. Department of Defense, February 1980, DTIC Number AD A100 404.

Synopsis: The Stoneman document defines the APSE as a layered system. The innermost layer is referred to as the Kernel APSE, or KAPSE. The KAPSE is machine-dependent and includes the database functions and other general operating system support functions. The next layer, the Minimal APSE, or MAPSE, consists of the minimal set of tools which can support the development of software. The outermost layer, the APSE, consists of tools and functions that are project dependent. In addition to providing guidance for APSE designers, the Stoneman document provides some evaluation criteria for APSEs.

4.2 HOUGHTON: A TAXONOMY OF TOOL FEATURES FOR THE Ada PROGRAMMING SUPPORT ENVIRONMENT (APSE)

Citations: [Houghton 1983] R.C., Houghton, Jr., "A Taxonomy of Tool Features for the Ada Programming Support Environment (APSE)," National Bureau of Standards, NBSIR-81-2625, February 1983.

Synopsis: This paper puts forth a taxonomic classification of APSE features. The features included satisfy the criteria that they are "within current technology" and are "oriented to the Ada language." The top two levels of the classification are as follows:

- Input
  - Subject
  - Control Input
- Function
  - Transformation
  - Management
  - Static Analysis
  - Dynamic Analysis
- Output
  - User Output
  - Machine Output

For each of the second-level elements above, a third-level list is given, and some discussion is provided. The paper includes the results of a survey in which the second and third-level elements under "Function" are each rated as "Required," "Important," or "Useful."

4.3 E&V REPORT: DoD APSE ANALYSIS

Citations: [Castor 1984] V.L. Castor, "DoD APSE Analysis Report, Draft Version 1.0," 31 August 1984, Appendix C of "Evaluation and Validation (E&V) Team Public Report", Air Force Wright Aeronautical Laboratories, November 1984, DTIC Number AD A153 609.

Synopsis: The DoD Ada Programming Support Environment (APSE) Analysis Document was prepared by the APSE Working Group (APSEWG) of the E&V Team. It contains a description and analysis of the Ada programming support environments developed by each of the armed services. The three environments analyzed were the Air Force's Ada Integrated Environment (AIE), the Army's Ada Language System (ALS), and the Navy's Ada Language System/Navy (ALS/N). The design documentation was used to determine the functionality contained in each programming environment. The functions were described in a taxonomy in order to determine the commonality and differences of each system. The taxonomy developed for this purpose was an expanded version of the function part of the taxonomy developed earlier by Houghton [@Houghton 1983]; see synopsis [4.2].

4.4 CLASSIFICATION SCHEMA/E&V TAXONOMY CHECKLISTS

Citations: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

Synopsis: The purpose of this document was to set forth a schema, or a framework, to be used in subsequent E&V documents, especially the E&V Reference Manual [@RM]. The Function Index of the schema was strongly influenced by earlier documents, such as Houghton's taxonomy [4.2], the DoD APSE Analysis Report [4.3], and the SEE tool features taxonomy [@Kean 1985]. The upper levels of the Function Index of the schema became the initial version of the Function Index of the Reference Manual. The lower levels were found to incorporate a large number of tool functions which could be evaluative in nature. These tool function features have been carried over into the Guidebook as capability assessment checklists. As a group, they are considered the Classification Schema Checklists.

Methods:

[Compilation Checklist	6.8;
Program Library Management Checklist	6.9;
Linking/Loading Checklist	7.2;
Import/Export Capabilities Checklist	7.3;
Debugging Capabilities Checklist	7.5;
Real-Time Analysis Capabilities Checklist	7.7;
Configuration Management Capabilities Checklist	11.1;
Electronic Mail Capabilities Checklist	15.3;
Text Editing Capabilities Checklist	16.1]

4.5 REQUIREMENTS FOR E&V

Citations: [Szymanski 1987] R. Szymanski, "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 2.0," 4 December 1986, Appendix D of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, September 1987, DTIC Number AD A196 164.

[Castor 1984] V.L. Castor, "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 1.0," 17 October 1984, Appendix B of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, November 1984, DTIC Number AD A153 609.

Synopsis: This document was prepared by the Requirements Working Group (REQWG) of the E&V Team. Its purpose is to set forth requirements on the E&V Project. It is intended for use by the E&V Team and by the E&V Project contractors in developing technology for the evaluation and validation of APSEs. However, its use in other E&V efforts is encouraged. The document contains three categories of requirements: (1) those on the E&V Team itself, (2) those on the E&V methods and procedures, and (3) those specifying what is to be evaluated or validated. See also the Tools and Aids Document, synopsis [4.6].

Version 1.0 of the document contains three questionnaires for assessing: command language interpreters, compilers, and configuration management tools.

Methods:

[Compiler Assessment Questionnaire	6.12;
Command Language Interpreter Assessment Questionnaire	9.3;
Configuration Management Assessment Questionnaire	11.3]

4.6 TOOLS AND AIDS FOR E&V

Citations: [Szymanski 1990] R. Szymanski, "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, December 1990, Appendix ?, DTIC Number to be assigned.

Synopsis: This document was prepared by the Requirements Working Group (REQWG) of the E&V Team. It identifies the community's E&V technology needs, provides definitions of those needs, and prioritizes them. The purpose of this document is to provide pertinent information to those agencies willing and able to fund the development of E&V technology. It reflects the E&V Requirements Document (see synopsis [4.5]) and views on the subject obtained from surveys conducted among the E&V Team and appropriate ARPANET-MILNet Interest Groups.

4.7 STARS-SEE OPERATIONAL CONCEPT DOCUMENT

Citations: [STARS 1985] "Proposed Version 001.0," STARS Joint Service Team for Software Engineering Environments, Stars Joint Program Office, October 1985.

Synopsis: The Software Technology for Adaptable, Reliable Systems—Software Engineering Environment (STARS-SEE) Operational Concept Document (OCD) presents requirements from the perspective of the STARS-SEE users. It represents a consensus among the Government agencies responsible for SEE development and support, STARS-SEE implementors, and potential users. Major sections of the document describe the STARS-SEE mission, operational and support environments, and system components and functions. The primary mission centers on the development, support, reuse, management, and control of mission critical software. The STARS-SEE system is defined to consist of the people, computers, software, and procedures needed to perform the mission.

Major topics discussed include:

- the types of users and associated software activities
- the function of the Integration and Compatibility Framework
- the capabilities required by the Information Storage and Retrieval System
- the functional capabilities of the SEE
- the SEE-user interaction
- the hardware and software characteristics of the computer system.

The functional capabilities address project planning and control, requirements specification and analysis, design specification and analysis, software prototyping and modeling, reusability, program generation and unit testing, integration testing, quality assurance, verification and validation, configuration management, software/hardware integration, post deployment software support, project communications, generation of documents, data collection, performance and productivity measurement, help and training for STARS-SEE users, the transition to and tailoring of the STARS-SEE, and knowledge engineering.

4.8 GRUND, ET AL.: KEY CHARACTERISTICS OF APSES

Citations: [Grund 1985] E.C. Grund, L.A. Hilliard, and K.A. Younger, "Key Characteristics of Ada Programming Support Environments," MITRE Corporation, ESD-TR-85-144, MTR-9590, July 1985, DTIC Number AD B096 137.

Synopsis: This document is intended to provide basic information about Ada Programming Support Environments for people concerned with the specification or selection of an APSE. Section 1 summarizes the STONEMAN APSE requirements. Section 2 describes desirable characteristics of APSEs in five areas: compilers, run-time environments, databases, configuration management tools, and editors. A short list of questions to ask in each area is included. Section 3 describes four Ada programming support products available or under development in early 1985 in terms of their capabilities in the same five areas.

4.9 Ada-EUROPE: SELECTING AN Ada ENVIRONMENT

Citations: [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986.

Synopsis: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@Buxton 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided.

The most significant aspect of this book is not that it provides all of the answers, but that it raises the proper questions. It provides a good companion guide for specifying, building, selecting, and using an environment. In addition, it provides a synopsis of the industry's best understanding of the environment issues to date.

The structure is represented by the table of contents of the guide, reproduced in part below.

Part A Host and Target Considerations
- 2. Underlying machine
- 3. Target machine

Part B Kernel
- 4. Database, schema and typing
- 5. Versions, configurations and history
- 6. Security and integrity
- 7. Language issues and run-time support
- 8. Interaction between programs

Part C Aids for Tool Building
- 9. Meta-tools and tool components

Part D Man-Machine Interaction
- 10. Administrative aspects
- 11. The user interface
- 12. Help, error and warning messages

Part E Tool Functions
- 13. Office automation aspects
- 14. Static analysis, compilation and the program library
- 15. Testing, debugging and dynamic analysis
- 16. Project and product management
- 17. Life cycle support

Part F Other Issues
- 18. Performance of the environment
- 19. Contractual matters

Methods:

4.10 MCDERMID AND RIPKEN: LIFE CYCLE SUPPORT IN THE Ada ENVIRONMENT

Citations: [McDermid 1984] J. McDermid and K. Ripken, "Life Cycle Support in the Ada Environment," Cambridge University Press, 1984.

Synopsis: This book contrasts its own approach to APSE development with that of the Stoneman report [@Buxton 1980]. Stoneman takes a bottom-up approach, starting with a kernel and minimal APSE (KAPSE and MAPSE), as a foundation for extensions to more powerful and better integrated environments. McDermid and Ripken follow a top-down approach by defining requirements for a coherent life-cycle methodology. They then describe a particular instance of a coherent methodology, as a combination of existing methods used in various life-cycle phases. This description becomes the basis for a definition of a "coherent APSE" that supports the entire life cycle.

The authors use a seven-phase life cycle and state requirements for each phase in terms of

- a system **representation** form
- a **transformation** method
- a **verification** activity.

Table 4.10-1 lists the names of the seven phases (each named for its principal output) and the methods selected for each.

Table 4.10-1 Example Coherent Methodology

PHASE (OUTPUT)	SELECTED METHOD
Requirements Expression	CORE
System Specification	A-7 Techniques
Abstract Functional Specification	A-7 Techniques
Module Specification	Ada and ANNA
Module Design	Ada and ANNA
Module Code	Ada and ANNA
Executable System	--

The authors are not completely satisfied with all of the methods chosen, and point out shortcomings in each case. They suggest the book be used as "a reference point for further work on APSE design and development." They stress that the coherence of the methods and ease of transition from one phase to the next is an important attribute. They also outline a phased development plan in which a larger scale APSE might be developed in the following three steps:

- a "Clerical Support APSE"
- a "V&V and Management Support APSE"
- a "Transformation Support APSE."

4.11 NOTKIN AND HABERMANN: SOFTWARE DEVELOPMENT ENVIRONMENT ISSUES AS RELATED TO Ada

Citations: [Notkin 1981] D.S. Notkin and A.N. Habermann, "Software Development Environment Issues as Related to Ada," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 107-133.

Synopsis: This paper addresses software development problems that arise in three areas: programming, system composition, and management. In each area traditional methods and tools are contrasted with a more integrated approach exemplified by an experimental environment named Gandalf.

"Programming issues are those that arise when a single programmer takes a program all the way from its specifications to a working program."

"System composition issues are those that arise when a system (or a version of a system) is built by integrating many programs into one." "The two basic problems in system composition are interface control and version control." Traditional methods use isolated tools "coordinated by memory...or scraps of paper."

"Management issues are those that arise when a group of more than one person develops and maintains a system over a period of time." Three problem categories are addressed: misunderstanding, lack of information, and conflict of interest. Traditionally, these problems have been handled by non-technical means. The problem with the management approach to a management environment is that the solution to human interaction difficulties is treated by the introduction of more human interaction.

Although this paper was not written as an example of E&V technology, the following list of environment software requirements (paraphrased from the paper) may be used as a high-level checklist:

- Concurrent multiple users must be supported
- An efficient implementation of Ada must be possible
- Efficient support for data base manipulations is needed
- A good file system is essential
- An extensible command language is needed.

It is also pointed out that the most important hardware requirement is that the software requirements listed above must be supported.

4.12 STENNING, ET AL.: THE Ada ENVIRONMENT: A PERSPECTIVE

Citations: [Stenning 1981] V. Stenning, T. Froggart, R. Gilbert, and E. Thomas, "The Ada Environment: a Perspective," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 36-46.

Synopsis: This paper discusses the objectives and the design of the Ada Programming Support Environment. It is strongly influenced by the United Kingdom Ministry of Defense Ada Support System Study, which was initiated by the MoD in January 1979. According to the paper, the DoD KAPSE/MAPSE/APSE approach is strongly recommended to achieve portability. The APSE should be designed to support a project throughout its life cycle. Furthermore, it should be an open-ended environment. This would allow for the user to extend or modify existing tools. A basic configuration control manager, a complete user interface, and a complete basic tool set are necessary to develop an Ada Environment which will improve program reliability, life-cycle program costs, and promote portability.

4.13 WEIDERMAN: EVALUATION OF Ada ENVIRONMENTS

Citations: [Weiderman 1987] N.H. Weiderman and A.N. Habermann, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, DTIC Number AD A180 905.

Synopsis: In response to the lack of available research about the selection of APSEs, the Software Engineering Institute (SEI) has developed a methodology to evaluate these environments. The methodology centers around the execution of several experiments in the environment to be evaluated. Several experiments have been developed in the following areas: System Management; Configuration Management/Version Control; Design and Code Development; Unit Testing and Debugging. The environments are evaluated in terms of functionality, performance, user interfaces, and system interfaces. The need for an evaluator to tailor an evaluation technique to a specific environment is addressed by the SEI study. The experiments that have been designed are generic experiments. The evaluator derives, or "instantiates," the environment-specific technique from the generic experiment. In the final phase of the evaluation, the results are analyzed. An advantage of the application of this methodology is that results can be compared from one environment to another. See also a paper describing an application of the SEI's method [@Gray 1987].

Methods:

[SEI Debugging Experiment	7.9;
SEI Unit Testing Experiment	8.2;
SEI Design Support Experiment	10.1;
SEI Configuration Management Experiment	11.2]

4.14 BARSTOW AND SHROBE: OBSERVATIONS ON INTERACTIVE PROGRAMMING ENVIRONMENTS

Citations: [Barstow 1981] D.R. Barstow and H.E. Shrobe, "Observations on Interactive Programming Environments," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 285-301.

Synopsis: This paper reviews key features of LISP-based environments and comments upon lessons learned from these environments and future directions. These environments encourage a "progressive enrichment" style of development rather than developments broken into distinct phases such as specification, implementation, and maintenance. The following set of lessons (described more fully in the paper) are concerned with the programmer's perception of the environment:

- It is important to be able to run an incomplete program.
- The user should be able to view the program from many different natural viewpoints, most of which are "structured" in nature.
- Intercommunication among tools is extremely important.
- The programmer should not be required to know the details of the particular language definition used in the current implementation.
- The environment's interface must be highly tuned to be as natural as possible for the human programmer.

Environment characteristics created with these lessons in mind "lead to the ultimate goal of a programming environment (which is to increase the ability of the programmer to communicate with the computer) by taking advantage of as many naturally occurring structures as possible."

4.15 HOUGHTON AND WALLACE: CHARACTERISTICS AND FUNCTIONS OF SOFTWARE ENGINEERING ENVIRONMENTS: AN OVERVIEW

Citations: [Houghton 1987] R.C. Houghton, Jr. and D.R. Wallace, "Characteristics and Functions of Software Engineering Environments: An Overview," ACM Software Engineering Notes, Vol. 12 Number 1, January 1987.

Synopsis: This paper provides a comprehensive discussion of software engineering environments in general, with no focus on Ada or any specific language. Some major topics discussed are:

- Environment Types and Life Cycle Relationships
- Integration
- Human Factors
- Analysis and Software Quality
- Support for Different Types of Users
- Support for Application
- Hardware Support
- Levels of Support.

In its concluding section, the paper stresses that software engineering environments should be viewed as systems that support broad categories of users and tasks throughout the full life cycle.

4.16 CAIS AND CAIS-A: DoD-STD-1838 AND MIL-STD-1838A

Citations: [DoD 1986] DoD-STD-1838, Common APSE Interface Set (CAIS), U.S. Department of Defense, 9 October 1986, DTIC Number AD A157 589.

[DoD 1989] "Common APSE Interface Set, Revision A," MIL-STD-1838A, April 1989, DTIC Number AD A157 589.

Synopsis: DoD-STD-1838, hereafter called CAIS, was developed by the KAPSE Interface Team (KIT) and the KAPSE Interface Team for Industry and Academia (KITIA) during the period from 1981 to 1986 as a first evolutionary step towards a full state-of-the-art common APSE interface standard.

The CAIS is designed to promote source-level portability of Ada programs, especially Ada software development tools. The goal of the CAIS is to promote interoperability (of database objects) and transportability (of APSE tools) of Ada software across Department of Defense (DoD) APSEs. See also the overview paper [@Oberndorf 1988].

CAIS-A is a set of Ada package interfaces designed to enhance the transportability of Ada Support Environment Tools. The scope of the CAIS-A includes the functionality affecting transportability that is needed by tools, but not provided by the language. The CAIS-A contains definitions for an entity management system for software engineering tools. The primitive entities defined allow for the manipulation of devices, files, and processes. CAIS-A is based on an entity-relationship approach and it allows the user to define entities, in a limited way, by means of a typing mechanism. CAIS-A also includes functionality to support tools requiring transaction processing, a rudimentary triggering mechanism, and explicit control over APSE distribution.

The CAIS-A was developed by SofTech under contract to Naval Ocean Systems Center. CAIS-A is a design enhancement to the existing CAIS (DoD-STD-1838), which was developed by the KIT and KITIA as a first evolutionary step towards a full, state-of-the-art interface standard. Designers view CAIS-A as the next step in that evolutionary process.

4.17 NISSEN, ET AL: GUIDELINES FOR Ada COMPILER SPECIFICATION AND SELECTION

Citations: [Nissen 1984] J.C.D. Nissen, B.A. Wichmann, et al., "Guidelines for Ada Compiler Specification and Selection," in *Ada: Language, Compilers and Bibliography*, ed. M.W.Rogers, Cambridge University Press, 1984.

Synopsis: Members of Ada-Europe produced this set of guidelines based upon a taxonomy of compiler features. Their caveat is clear: "The relative value of information about different features of the compiler is a matter of judgment and circumstance.... It is the reader's responsibility to weigh each factor according to his requirements. No liability of whatever kind shall be carried by the authors."

The taxonomy is represented by the table of contents of the guide, reproduced in part below.

2. Host and target
3. Language-related issues
4. User-interfacing and facilities
    - 4.1 Compiler invocation and listing management
    - 4.2 Compilation options
    - 4.3 Other features
    - 4.4 Errors and warnings
    - 4.5 Other software supplied
    - 4.6 Compilation management
5. Performance and capacity
    - 5.1 Host performance and capacity
    - 5.2 Target code performance
6. Compiler and run-time interfacing
    - 6.1 Compiler issues
    - 6.2 Run-time system issues
7. Retargetting and rehosting
    - 7.1 Introduction and definitions
    - 7.2 Retargetting
    - 7.3 Rehosting
8. Contractual matters
9. Validation

Chapter 2, Host and Target, briefly treats compiler configuration issues, and provides a questionnaire [Host and Target Questionnaire 5.1].

Chapter 3, Language-related issues, extracts from the Ada language reference manual [@DoD 1983] those features explicitly allowed to vary based upon machine specific characteristics [Machine-specific Characteristics Questionnaire 5.2].

Methods:

[Host and Target Questionnaire 5.1;
Machine-specific Characteristics Questionnaire 5.2;
Ada Compiler Specification and Selection Questionnaires 6.18]

4.18 WEIDERMAN: COMPILER EVALUATION AND SELECTION

Citations: [Weiderman 1989] N.H. Weiderman, "Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0," Software Engineering Institute, CMU/SEI-89-TR-13, March 1989, DTIC Number AD A207 717.

Synopsis: The following Abstract is quoted directly from the cited document:

> The evaluation and selection of an Ada compilation system for a project is a complex and costly process. Failure to thoroughly evaluate an Ada compilation system for a particular user application will increase project risk and may result in cost and schedule overruns. The purpose of this handbook is to convince the reader of the difficulty and importance of evaluating an Ada compilation system (even when there is no freedom of choice). The handbook describes the dimensions along which a compilation system should be evaluated, enumerates some of the criteria that should be considered along each dimension, and provides guidance with respect to a strategy for evaluation. The handbook does not provide a cookbook for evaluation and selection. Nor does it provide information on specific compilation systems or compare different compilation systems. Rather it serves as a reference document to inform users of the options available when evaluating and selecting an Ada compilation system.

The chapter headings are as follows:

1. Introduction
2. Common Questions
3. Compiler Validation and Evaluation
4. Practical Issues of Selecting an Ada Compiler
5. Compile/Link-Time Issues
6. Execution-Time Issues
7. Support Tool Issues
8. Benchmarking Issues
9. Test Suites and Other Available Technology

In Chapters 4 through 8 there are a number of lists (some annotated) of criteria, factors, features, and questions to be asked. Some of these are synopsized in, or have influenced, later sections of this document, as indicated below.

Methods:

[Weiderman: Compiler Evaluation Lists 6.13;
Vendor Evaluation Questionnaire 5.5]

4.19 Ada PERFORMANCE ISSUES (PIWG SPECIAL EDITION)

Citations: [PIWG 1990] "Ada Performance Issues," Ada Letters, Special Edition from SIGAda, the ACM Special Interest Group on Ada Performance Issues Working Group, Vol. X, Number 3, Winter 1990.

Synopsis: This document, which was prepared by the Performance Issues Working Group (PIWG) of ACM/SIGAda, has three major parts: (I) Preface, (II) Rationale, and (III) Results. The Preface introduces the general topic of Ada performance assessment and distinguishes between language issues, compile-time issues, execution-time issues, and performance measurement issues. The Rationale provides a collection of chapters and invited papers covering a spectrum of topics such as benchmarking philosophy and taxonomy, the time problem, the space problem, parallel and distributed issues, optimization, and the PIWG's own measurement methodology. The Results part of the document presents a detailed, tabular summary of performance measurements generated by a number of PIWG volunteers and vendors who have used the PIWG test suite. Several of the items in Part II, as well as a further discussion of the PIWG test suite itself, are summarized in later sections of this Guidebook, as listed below.

Methods:

[PIWG Benchmark Tests	6.4;
Hartsone Synthetic Tests	6.15;
Ada Compiler Performance Test Suite (ACPS)	6.16;
Performance of Parallel Ada	12.1]

4.20 SOFTWARE TOOL EVALUATION MODEL (STEM)

Citations: [STSC 1990] "STSC Strategy," USAF STSC - HQ USAF/SC Joint Software Conference Proceedings, Salt Lake City, 23-26 April 1990.

[Berk 1990] K.J. Berk and R.P. Hanrahan, "Evaluating Tools and Environments Concept," Proceedings of the IEEE 1990 National Aerospace and Electronics Conference, NAECON 1990, May 1990, Volume 2, pp. 658-663.

Synopsis: The Software Technology Support Center (STSC) at Hill AFB "will provide Air Force users with a focal point for current information about software tools, development, maintenance environments, and methods. STSC will provide this information to all requesting Air Force organizations. This information will cover the software acquisition, development, and maintenance life-cycle." Several of the activities of the STSC are centered around the Software Tool Evaluation Model (STEM). "STEM is a framework to coordinate STSC's activities, tools, methods, and environments. STEM will be a reference or an ideal to which actual software tools and environments can be compared. STEM will also provide a categorization scheme for classifying individual software tools. These tool categories will be important in creating the computer data bases that contain user-accessible information about tools. Finally, STEM will be a standard by which specifications can be written for defining and acquiring new tools and software engineering environments." An early activity of STSC is the preparation and distributution of "a data base (called Toolbox/PC) that contains basic information about selected available tools and provides a mechanism for collecting user information about additional tools and user needs."

There have been STSC-sponsored efforts to assess tools in three categories (Test, Requirements and Design, and Documentation) as indicated below.

Methods:

[STEM/SAIC Test Tools Evaluation	8.3;
STEM/Draper Req/Design Tools Evaluation	10.6;
STEM/TRW Documentation Tools Evaluation	16.6]

4.21 INFORMATION RESOURCE DICTIONARY SYSTEM (IRDS)

Citations: [ANSI 1988] "Information Resource Dictionary System (IRDS)," American National Standard for Information Systems, ANSI X3.138-1988, American National Standards Institute, Inc., New York, NY, 1988.

[Law 1988] M.H. Law, "Guide to Information Resource Dictionary System Applications: General Concepts and Strategic Systems Planning," NBS Special Publication 500-152, National Bureau of Standards, Gaithersburg, MD, April 1988.

[FIPS 1989] "Information Resource Dictionary System (IRDS)," Federal Information Processing Standard Publication, FIPS PUB 156, National Institute of Standards and Technology, Gaithersburg, MD, April 5, 1989.

Synopsis: The Information Resource Dictionary System (IRDS) is an American National Standards Institute (ANSI) standard approved in 1988. It has also been adopted as a Federal Information Processing Standard (FIPS) in 1989. The standard establishes the requirements for an IRDS, a data repository which can be used to control, describe, protect, document, and facilitate use of an installation's information resources. Work on the standard began in 1980.

The IRDS standard is made up of seven modules:

- Core standard
- Basic functional schema
- IRDS security
- Extensible life cycle phase facility
- IRDS procedures
- Application program interface
- Services interface.

The core standard provides a number of useful facilities: entity naming conventions, a command or panel user interface, import-export interface, version naming, life cycle control, views, and more. The basic functional schema is a data model provided to support an organization's information processing needs. The IRDS security facilities provides the ability to define access controls for dictionary development, maintenance, and use on both a global and entity basis. The extensible life cycle phase facility extends the life cycle capabilities of the core by implementing integrity rules and customization features to control the movement of entities through the life cycle or, in other words, aids in the transition from one development phase to the next. The IRDS procedures provide the

ability to define, store, maintain, and execute procedures. The application program interface provides access to the IRDS from traditional procedural languages such as COBOL or FORTRAN for the purpose of building tools that use the IRD data. The services interface is really a low-level processor-to-processor interface that is still being defined.

5. GENERAL PURPOSE ASSESSORS

These assessors examine the characteristics of APSEs or APSE components that are independent of the function(s) implemented by the APSE or component. These assessors are therefore applicable to all components within the APSE since they are used to determine qualities such as usability, expandability and flexibility, interoperability, reusability, and transportability in a general sense.

5.1 HOST AND TARGET QUESTIONNAIRE

Purpose: Evaluation of tools relative to host and target configurations.

[@RM: Rehostability 6.4.26;
@RM: Retargetability 6.4.28]

Primary References: [Nissen 1984] J.C.D. Nissen, B.A. Wichmann, et al., "Guidelines for Ada Compiler Specification and Selection," in *Ada: Language, Compilers and Bibliography*, ed. M.W. Rogers, Cambridge University Press, 1984.

[Nissen, et al.: Guidelines For Ada Compiler Specification And Selection 4.17]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire

Inputs: Questionnaire and tool documentation.

Process: Fill in the appropriate answers in the following questionnaire.

a) Host configuration(s) required
b) Host operating system(s) required
c) Target configuration(s) supported
d) Target operating system(s) supported
e) APSE(s) supported, if applicable
f) Host-target communication supported
    i) program loading
    ii) program execution and debugging.

Outputs: A completed list which characterizes the tool relative to host-target issues.

5.2 MACHINE-SPECIFIC CHARACTERISTICS QUESTIONNAIRE

Purpose: Evaluation of tools relative to machine-specific characteristics.

[@RM: Rehostability 6.4.26;
@RM: Retargetability 6.4.28]

Primary References: [Nissen 1984] J.C.D. Nissen, B.A. Wichmann, et al., "Guidelines for Ada Compiler Specification and Selection," in *Ada: Language, Compilers and Bibliography*, ed. M.W. Rogers, Cambridge University Press, 1984.

[Nissen, et al.: Guidelines For Ada Compiler Specification And Selection 4.17]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire.
Inputs: Questionnaire and tool documentation.
Process: Fill in the appropriate answers in the following questionnaire.

[@DoD 1983: Lexical Elements 2.]

- [@DoD 1983: 2.1] Character set of the host and target
- [@DoD 1983: 2.2] Maximum number of characters on a line of the host and target
- [@DoD 1983: 2.3, 2.4] Is the maximum character length of an identifier or numerical literal restricted other than by line length
- [@DoD 1983: 2.8, F.] The form, allowed place, and effect of every implementation-defined pragma

[@DoD 1983: Declarations and Types 3.]

- [@DoD 1983: 3.2.1] The effect of using uninitialized variables—does the compiler flag or reject program that depends upon such variables
- [@DoD 1983: 3.5.1] The maximum number of elements in an enumeration type
- [@DoD 1983: 3.5.4] The values of:
  - INTEGER'FIRST
  - SHORT_INTEGER'FIRST
  - LONG_INTEGER'FIRST
  - INTEGER'LAST
  - SHORT_INTEGER'LAST
  - LONG_INTEGER'LAST
- [@DoD 1983: 3.5.8] The values of:
  - FLOAT'DIGITS
  - SHORT_FLOAT'DIGITS
  - LONG_FLOAT'DIGITS

[@DoD 1983: Names and Expressions 4.]

- [@DoD 1983: 4.10] Is there a limit on the range of universal values which

exceeds the capacity of the compiler
- [@DoD 1983: 4.10] Is there a limit on the accuracy real universal expressions

[@DoD 1983: Tasks 9.]
- [@DoD 1983: 9.6] The values of:
  - DURATION'DELTA
  - DURATION'SMALL
  - DURATION'FIRST
  - DURATION'LAST
- [@DoD 1983: 9.8] The values of:
  - PRIORITY'FIRST
  - PRIORITY'LAST
- [@DoD 1983: 9.11] The restrictions on shared variables

[@DoD 1983: Program Structure and Compilation Issues 10.]
- [@DoD 1983: 10.1] Initiation, communication with, and restrictions on the main program
- [@DoD 1983: 10.5] When tasks initiated in imported library units will terminate

[@DoD 1983: Exceptions 11.]
- [@DoD 1983: 11.1] Conditions under which these exceptions are raised:
  - NUMERIC_ERROR
  - PROGRAM_ERROR
  - STORAGE_ERROR

[@DoD 1983: Representation Clauses and Implementation-Dependent Features 13.]
- [@DoD 1983: 13.4, F.] The list of all restrictions on representation clauses
- [@DoD 1983: 13.1, F.] The conventions used for any system generated name denoting system dependent components
- [@DoD 1983: 13.5, F.] The interpretation of expressions that appear in address clauses, including those for interrupts
- [@DoD 1983: 13.7] The specification of package SYSTEM; which includes the values of:
  - MIN_INT
  - MAX_INT
  - MAX_DIGITS
  - MAX_MANTISSA
  - FINE_DELTA
  - TICK

[@DoD 1983: 13.7.3] For a pre-defined floating point type F, the value of:

- F'MACHINE_ROUNDS
- F'MACHINE_RADIX
- F'MACHINE_MANTISSA
- F'MACHINE_EMAX
- F'MACHINE_EMIN
- F'MACHINE_OVERFLOWS

- [@DoD 1983: 13.7.3] The values outside the range of safe numbers for real types
- [@DoD 1983: 13.10.1] Any restriction on UNCHECKED_DEALLOCATION
- [@DoD 1983: 13.10.2, F.] Any restriction on UNCHECKED_CONVERSION

[@DoD 1983: Input-Output 14]

- [@DoD 1983: 14., F.] Any implementation-dependent characteristics of the input-output packages
- [@DoD 1983: Implementation-Dependent Features F.]
  - [@DoD 1983: F.] The name and type of every implementation-dependent attribute

Outputs: A completed list which characterizes the tool relative to machine dependencies.

5.3 RADC SOFTWARE QUALITY METRIC WORKSHEETS

Purpose: "The purpose of this guidebook is to provide a comprehensive set of procedures and techniques to enable data collection personnel to apply quality metrics to software products and to evaluate the achieved quality levels." The focus of the RADC report is planning and designing quality into application software throughout the software life cycle. However, many of the questions on the worksheets are equally relevant to systems and support software and may be rephrased to address software that is already in use as opposed to software under development.

[@RM: Quality Assessment	7.2.2.8,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Application Independence	6.4.3,
		@RM: Augmentability	6.4.4,
		@RM: Autonomy	6.4.5,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Consistency	6.4.10,
		@RM: Distributedr s	6.4.12,
		@RM: Documentation Quality	6.4.13,
		@RM: Functional Overlap	6.4.14,
		@RM: Functional Scope	6.4.15,
		@RM: Generality	6.4.16,
		@RM: Modularity	6.4.20,
		@RM: Operability	6.4.21,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Reconfigurability	6.4.25,
		@RM: Rehostability	6.4.26,
		@RM: Retargetability	6.4.28,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: Simplicity	6.4.30,
		@RM: Storage Effectiveness	6.4.32,
		@RM: System Accessibility	6.4.33,
		@RM: System Clarity	6.4.34,
		@RM: System Compatibility	6.4.35,
		@RM: Traceability	6.4.36,
		@RM: Training	6.4.37,
		@RM: Virtuality	6.4.39,
		@RM: Visibility	6.4.40);

@RM: Quality Measurement	7.3.1.9,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Application Independence	6.4.3,
		@RM: Augmentability	6.4.4,
		@RM: Autonomy	6.4.5,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Consistency	6.4.10,
		@RM: Distributedness	6.4.12,
		@RM: Documentation Quality	6.4.13,
		@RM: Functional Overlap	6.4.14,
		@RM: Functional Scope	6.4.15,
		@RM: Generality	6.4.16,
		@RM: Modularity	6.4.20,
		@RM: Operability	6.4.21,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Reconfigurability	6.4.25,
		@RM: Rehostability	6.4.26,
		@RM: Retargetability	6.4.28,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: Simplicity	6.4.30,
		@RM: Storage Effectiveness	6.4.32,
		@RM: System Accessibility	6.4.33,
		@RM: System Clarity	6.4.34,
		@RM: System Compatibility	6.4.35,
		@RM: Traceability	6.4.36,
		@RM: Training	6.4.37,
		@RM: Virtuality	6.4.39,
		@RM: Visibility	6.4.40)]

Primary References: [Bowen 1985] T.P. Bowen, G.B. Wigle, and J.T. Tsai, "Specification of Software Quality Attributes Software Quality Evaluation Guidebook," Rome Air Development Center, Griffiss AFB, RADC-TR-85-37, Volume III (of three), February 1985, Appendix A, DTIC Number AD A153 990.

Vendors/Agents: [RADC, Boeing Aerospace]

Method: Questionnaire/Worksheet.

Inputs: Worksheet, tool, and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed worksheet.

5.4 SEI ASSESSMENT OF SOFTWARE ENGINEERING TOOLS

Purpose: The guide provides and discusses a set of standard questions that a potential user may ask about a tool, given that different users will interpret the answers in different ways and attach different degrees of importance to them. The questions are grouped according to the following aspects: 1) Ease of use, 2) Power, 3) Robustness, 4) Functionality, 5) Ease of insertion, and 6) Quality of commercial support. The first four sections are mainly of concern to the actual user of the tool; the last two are of concern to the management of the project that contemplates acquiring the tool.

[@RM: Anomaly Management	6.4.2;
@RM: Augmentability	6.4.4;
@RM: Capacity	6.4.6;
@RM: Commonality	6.4.7;
@RM: Completeness	6.4.9;
@RM: Consistency	6.4.10;
@RM: Functional Overlap	6.4.14;
@RM: Granularity	6.4.17;
@RM: Maturity	6.4.19;
@RM: Operability	6.4.21;
@RM: Power	6.4.22;
@RM: Processing Effectiveness	6.4.23;
@RM: Proprietary Rights	6.4.24;
@RM: Rehostability	6.4.26;
@RM: Simplicity	6.4.30;
@RM: Storage Effectiveness	6.4.32;
@RM: System Accessibility	6.4.33;
@RM: System Compatibility	6.4.35;
@RM: Traceability	6.4.36;
@RM: Training	6.4.37;
@RM: Vendor Support	6.4.38;
@RM: Visibility	6.4.40]

Primary References: [Firth 1987] R. Firth, V. Mosley, R. Pethia, L. Roberts, W. Wood, "A Guide to the Classification and Assessment of Software Engineering Tools," Software Engineering Institute, Technical Report, CMU/SEI-87-TR-10, August 1987, DTIC Number AD A182 895.

Vendors/Agents: [SEI]

Method: Questionnaire.

Inputs: Questionnaire, tool, and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed questionnaire.

5.5 VENDOR EVALUATION QUESTIONNAIRE

Purpose: The purpose of this questionnaire is to provide an overview of the characteristics and policies of a vendor. The questionnaire appears in full in Section 4.8 of the reference cited below. In its current form it applies specifically to Ada compilation system vendors, but most of the questions apply equally well to tool vendors in general. Figure 5.5-1 simply lists the titles of the 11 subdivisions of the questionnaire. The questionnaire itself provides 2 to 15 questions under these titles.

[@RM: Cost	6.4.11;
@RM: Documentation Quality	6.4.13;
@RM: Maturity	6.4.19;
@RM: Proprietary Rights	6.4.24;
@RM: Training	6.4.37;
@RM: Vendor Support	6.4.38]

Primary References: [Weiderman 1989] N.H. Weiderman, "Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0," Software Engineering Institute, CMU/SEI-89-TR-13, March 1989, DTIC Number AD A207 717.

[Weiderman: Compiler Evaluation and Selection 4.18]

Vendors/Agents: [SEI]

Method: Questionnaire.
Inputs: Blank characterization form and tool documentation.
Process: Gather data and fill in form.
Outputs: Completed vendor characterization form.

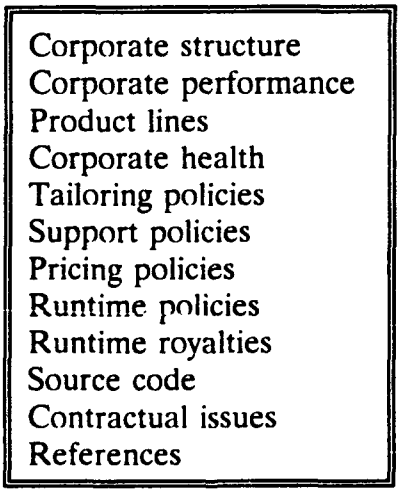

Figure 5.5-1 Vendor Characterization Form Categories

5.6 REQUIRED CONFIGURATION QUESTIONNAIRE

Purpose: Assess the required configuration for using a software product. The questionnaire covers the recommended as well as the required configuration since using a product with the minimum required configuration may result in performance which is unacceptable to a user.

[@RM: Rehostability 6.4.26;
@RM: Required Configuration 6.4.27;
@RM: System Compatibility 6.4.35]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 5.6-1) and the product documentation.

Process: Answer questions based on the documentation or by asking the product vendor.

Outputs: Completed questionnaire which describes the resources required to run the product.

Host System

What host computer(s) or chip(s) does the product run on?

What operating system(s) (including version and release) does the product run on?

- What is the minimum and recommended configuration of the operating system (parameter settings)?

Main Memory

What is the minimum required RAM (both physical, including expanded or extended, and virtual memory) to run the product?

- What is the recommended RAM to run the product?

- If expanded or extended memory is used, what type(s)?

How do the RAM requirements change with the size of the input data file(s)?

Secondary Memory

How much space do the executable(s), object file(s), and runtime system take?

How much space do typical input data file(s) take?

- How does the space vary with the size of the input data file(s)?

How much space do typical output data file(s) take?

- How does the space vary with the size of the output data file(s)?

Peripherals (Disk, Monitor, Printers, Plotters, Mouse, etc.)

What are the required peripherals?

What are the recommended peripherals?

Other Software (APSE, CAIS, Runtime, DBMS, Window Manager, etc.)

What are the other required software products?

What are the other recommended software products?

Figure 5.6-1 Required Configuration Questionnaire

5.7 COST QUESTIONNAIRE

Purpose: Assess the costs of acquiring, running, and supporting a software product.

[@RM: Cost 6.4.11;
@RM: Training 6.4.37;
@RM: Vendor Support 6.4.38]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 5.7-1) and the product pricing information.

Process: Answer questions based on the pricing information or by asking the product vendor.

Outputs: Completed questionnaire which describes the costs of acquiring, running, and supporting the product.

Product

What is the single copy price?
Are there discounts for volume purchases?
- Does the entire quantity have to be purchased all at once or does the vendor track purchases by organization and automatically apply the discount when the required volume has been reached?

What is the price of a site license?
What is the cost of leasing the product?
- Is the lease perpetual or fixed term?
- If fixed term, what are the renewal terms?

How does the cost depend on characteristics of the host machine?
- What are the costs to add more users or workstations?
- What are the costs to move to a different host machine?

Are there discounts for government organizations?
Are there discounts for academic institutions?
Can the product be returned during some specified period for a full refund?
If available, what is the cost of purchasing the source code?

Maintenance

What is the one year maintenance cost?
- What does it provide in terms of:
  - Support?
  - Bug Fixes?
  - Upgrades?
  - Documentation Updates?

What happens in the event maintenance has been dropped?

What is the cost of hot line support?
- Is there an 800 number?

If there is an electronic bulletin board for problem reports and resolutions, what is the subscription fee?
What are the costs of making application-specific changes?

Training

What is the cost of:
- On-site training?
- Seminars?
- Video training courses?
- Computer based training?
- Users' group membership?
- Newsletter subscription?
- In-house consultants?

Are there training credits offered with the purchase of the product?
- How much are they worth?
- What can they be used for?
- How long are they good for?

Documentation

What is the price of:
- Installation guide?
- Users' guide?
- Reference manual?
- Interface manual?
- Quick reference card?
- Keyboard template?

Miscellaneous

If the product requires the purchase of other hardware or software, what does that cost?

What is the cost of installing the product?
What is the cost of running the product?
- Cost per run or session?
- Runs or sessions per day, week, month, or year?

What is the cost of supporting the product?
- Cost of computer resources?
- Cost of operations or support personnel?

Figure 5.7-1 Cost Questionnaire

5.8 MATURITY QUESTIONNAIRE

Purpose: Assess the maturity of a software product.

[@RM: Maturity 6.4.19;
@RM: Vendor Support 6.4.38]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 5.8-1) and the product historical information.

Process: Answer questions based on the historical information or by asking the product vendor.

Outputs: Completed questionnaire which describes the maturity of the product.

Product History

When was the product first released?
What is the current version and release?
What is the frequency of new versions and releases?
What are the procedures for testing new versions and releases?
- Alpha testing?

- Beta testing?

User Community

How many active users of the product?
- Will the vendor supply references?
- What applications has the product been used on?

Is the vendor willing to share problem reports and resolutions with the users?
Is there a users' group?
- How often do they meet?

Product Evaluation

How has the product been rated in the literature?

Are there independent evaluations of the product available?

Figure 5.8-1 Maturity Questionnaire

5.9 LICENSING ISSUES QUESTIONNAIRE

Purpose: Assess the licensing agreement for a software product.

[@RM: Proprietary Rights 6.4.24;
@RM: Vendor Support 6.4.38]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 5.9-1) and the product licensing agreement.

Process: Answer questions based on the licensing agreement or by asking the product supplier.

Outputs: Completed questionnaire which describes the licensing agreement for a product.

What, specifically, is being purchased, leased, or otherwise acquired? - Hardware? - Software? - Utilities? - Documentation? - If lease, is it perpetual or fixed term? - If fixed term, what are the renewal terms?
Is any part of the product "free" or shareware? - What are the restrictions and/or obligations in using the product?
What, specifically, is covered by patents, copyrights, trademarks, or agreements? - Is any of the product copy protected? - Can the user make backup copies for protection? - Can the documentation be copied?
Are there limitations on the number of users or workstations using the product?
Can the product be moved to a different machine or does it have to stay on a specific machine?

Figure 5.9-1 Licensing Issues Questionnaire

Is the source code available? - If so, what are the licensing terms? - If not, can it be put into escrow to protect the user in the event that the supplier goes out of business? - Who holds the rights to the user-modified source code? - What is the effect on the maintenance agreement?
What is the supplier's obligation to correct deficiencies?
What is the supplier's obligation to maintain upward compatibility across new versions or releases?
What is the supplier's obligation to provide interface information to the user? - What is the supplier's obligation to maintain fixed syntax and semantics of the product's interfaces?
What are the supplier's rights to: - Runtime versions of the product (no development tools)? - Objects generated by the product? - What are the procedures to account for and collect the royalties?
What are the user's rights to the executables and/or source code in the event that: - The supplier goes out of business? - The supplier is bought out by another company? - The supplier discontinues the product? - The supplier issues a new version or release?
What obligations of the supplier to third parties are inherited by the user?
What exceptional conditions and penalty clauses are in the agreement?

Figure 5.9-1 Licensing Issues Questionnaire (Continued)

5.10 SOFTWARE PRODUCTION VEHICLE(S) QUESTIONNAIRE

Purpose: Assess the software production vehicle(s) of a software product.

[@RM: Software Production Vehicle(s) 6.4.31]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 5.10-1) and the product specification data.

Process: Answer questions based on the specifications or by asking the product vendor.

Outputs: Completed questionnaire which describes the software production vehicle(s) of a product.

Requirements/Design

What front end methodologies are used to develop the product?

What front end software tools are used to develop the product?

Coding

What source languages are used to implement the product?

- Are the versions based on standards?

Is the source code automatically generated from the front end tools?

- What code generator(s) are used?

What compiler (assembler, interpreter) version is used?

- What parameters are set during compilation?

What is the host computer and operating system (including models and versions)?

- Is the development platform the same as the product host?
  - If not, what is the development computer and operating system?

What linker version is used?

- What parameters are set during linking?

Testing and Evaluation

What testing techniques are used?

- Static analysis?

- Dynamic analysis:
  - Structural coverage?
  - Domain/Path coverage?
  - Symbolic testing?
  - Mutation analysis?
  - Functional testing?
  - Random testing?

What release testing procedures are used?

- Alpha testing?

- Beta testing?

What testing tools are used?

User Feedback

What mechanisms are there for collecting user feedback?

What mechanisms are there for evaluating user feedback?

What mechanisms are there for responding to user feedback?

Configuration Management

What procedures are there for managing the configuration of the product?

What tools are there for managing the configuration of the product?

Figure 5.10-1 Software Production Vehicle(s) Questionnaire

5.11 CHARACTERISTICS OF INTEGRABLE SOFTWARE TOOLS

Purpose: Assessment of the characteristics of software tools that allow them to be easily integrated into a software development environment (SDE). "Since we cannot fully enumerate all of the SDEs in which a tool will eventually be included (such a list is continually changing), a very important issue is the characteristics a tool should have such that it will be easy to add it to integrated environments at any time."

[@RM: Augmentability	6.4.4;
@RM: Commonality	6.4.7;
@RM: Communication Effectiveness	6.4.8;
@RM: Consistency	6.4.10;
@RM: Granularity	6.4.17;
@RM: Integration	6.4.18;
@RM: Operability	6.4.21;
@RM: Proprietary Rights	6.4.24;
@RM: Rehostability	6.4.26;
@RM: Software Production Vehicle(s)	6.4.31;
@RM: System Compatibility	6.4.35;
@RM: Virtuality	6.4.39]

Primary References: [Nejmeh 1989] B.A. Nejmeh, "Characteristics of Integrable Software Tools," Software Productivity Consortium, INTEG_S/W_TOOLS-89036-N, Version 1.0, 23 May 1989.

Vendors/Agents: [SPC, INSTEP]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 5.11-1) and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed questionnaire.

Location Independence
Can the location of data a tool produces be readily changed by a system administrator without affecting the tool?
Can the location of the binaries of the tool be readily changed by a system administrator without affecting the tool?

Non-Assuming
Does the tool make unnecessary assumptions about the network? For example:
- A copy of its binaries will reside on each node
- Where printers are plugged in
- What kinds of backup devices are available
- What and where system processes are running

Non-Interfering
Is the environment free from any side effects of the tool, such as stolen CPU cycles or locked resources?

Tool Fragment Invocation
Is it possible to invoke every tool fragment (component) outside the native user interface of the tool?
If so, how:
- Command line?
- Function library?

Data Import
Does the tool support a data import capability (can it read and process data produced by a foreign tool)?
If so, how:
- Published type definitions and formats?
- Access programs?

Data Export
Does the tool support a data export capability (can it produce data that can be read and processed by a foreign tool)?
If so, how:
- Published type definitions and formats?
- Access programs?

Object Schema Extension
Is it possible for a user to define additional properties for an object and set/get the value(s) of the properties?
Can the tool display the values of such properties?

Contextual Information Exportation
Is it possible, from within the tool, to export information (data) about selected objects?

Operation Extension
Is it possible, once the context within a tool is established, for a user-supplied operation (program) to be executed on the established contextual data?

X Window System
Does the tool use the X Window System for its user interface?

Message Modification
Is it possible to modify the messages that the tool produces:
- Error messages?
- On-line help?

Hardware Transparency
Are all machine versions currently running the same release version?
Does the tool operate the same on different hardware platforms?
- Are the differences documented?
- Does the tool operate the same as other tools on the same hardware platform?

Reasonable Licensing Scheme
Does the tool conform to a standard licensing scheme (such as the Network Computing Forum scheme)?
If so, how:
- Floating keys?
- Pay per usage?

Figure 5.11-1 Characteristics of Integrable Tools Questionnaire

5.12 SERC: A FRAMEWORK FOR ANALYZING USER INTERFACES

Purpose: "A framework is presented for analyzing the capabilities of user interfaces. The different features that describe the design space of user interfaces are discussed along with the consequences of supporting them. The framework is used to analyze current interaction styles based on forms, teletype input/output, text editing, graphical display, and structure editing."

[@RM: Anomaly Management	6.4.2;
@RM: Consistency	6.4.10;
@RM: Distributedness	6.4.12;
@RM: Operability	6.4.21;
@RM: Power	6.4.22;
@RM: Processing Effectiveness	6.4.23]

Primary References: [Dewan 1988] P. Dewan, "A Framework for Analyzing User Interfaces," Software Engineering Research Center, Purdue University, SERC-TR-11-P, 24 May 1988.

Vendors/Agents: [SERC]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 5.12-1), tool, and tool documentation.

Process: Answer the questions based on running the tool, reading the tool documentation, or asking the tool vendor.

Outputs: Completed questionnaire.

Automatic/Manual Updates

Does the tool/APSE automatically update the display to show new values of objects or does the tool require the user to request the update via commands or both? (Manual updates may be desired if the tool manipulates a large number of objects which cannot all be shown on a single display.)

Is the mode of update appropriate to the type of interaction?

If both modes of update are supported, is the mode user-selectable or does the tool select the mode based on the specific interaction?

In-Place/Script Updates

Does the tool/APSE update the display in-place or does the tool update the display via a script or both? (In-place updates reduces display clutter by not showing redundant or possibly unimportant old information, while script updates show the history of the interaction.)

Is the type of update appropriate to the type of interaction?

If both types of update are supported, is the type user-selectable or does the tool select the type based on the specific interaction?

Graphical/Textual Presentations

Does the tool/APSE provide purely textual, purely graphical, or combined graphical-textual representation of objects?

Is the type of presentation appropriate to the type of interaction?

If both types of presentation are supported, is the type user-selectable or does the tool select the type based on the specific interaction?

Viewing Information at Various Levels of Detail

Does the tool/APSE allow the user to view objects at various levels of detail via compress and expand commands?

If so, does the tool replace one view with the other, show both views simultaneously, or both?

Is the technique appropriate to the type of interaction?

If both techniques are supported, is the type user-selectable or does the tool select the type based on the specific interaction?

Figure 5.12-1 Analyzing User Interfaces Questionnaire

Selection/Naming of Objects and Operations

How does the user specify the object(s) to be manipulated and the operation(s) to be performed by the tool/APSE; by naming them, by selecting them from a list, or both? (Selection saves the user from the task of knowing and entering the names of objects and operations, but may require the user to change views. Selection is preferable when several selections can be made before changing views.)

- Is the specification approach appropriate to the type of interaction?
- If both specification approaches are supported, is the type user-selectable or does the tool select the type based on the specific interaction?
- If the named specification approach is supported, does the tool support abbreviated forms?
  - If yes, are the abbreviated forms fixed or arbitrary?

Uniform/Non-Uniform Invocation of Operations

Does the tool/APSE support a strictly uniform operation invocation scheme (i.e., prefix, infix, or postfix notation), a non-uniform scheme, or a flexible scheme? (Uniform operation invocation schemes are generally easier to learn, but may be too rigid or non-intuitive in certain contexts.)

- Is the scheme appropriate to the type of interaction?
- If a flexible scheme is supported, is the scheme user-selectable or does the tool select the scheme based on the specific interaction?

Uniform/Non-Uniform Naming of Operations

Does the tool/APSE provide uniform naming of operations for similar (identical) operations performed on different types of objects?

Automatic/Manual Propagation of Changes

Does the tool/APSE automatically propagate changes to related objects or does the tool require the user to propagate the changes manually or both? (Automatic propagation is more powerful since the data is consistently maintained, but it may be expensive to recover from an erroneous transaction.)

- Is the mode of propagation appropriate to the type of interaction?
- If both modes of propagation are supported, is the mode user-selectable or does the tool select the mode based on the specific interaction?

Direct Manipulation

Does the tool/APSE support the modification of objects by direct manipulation, i.e., by editing its presentation rather than forcing the user to invoke a series of commands? (Direct Manipulation prevents errors and saves the user from having to confirm changes.)

- If so, does the tool do so for all objects or only some?

User-Driven/Program-Driven Execution of Operations

Does the tool/APSE support program-driven or user-driven execution of operations or both? (User-driven execution allows the user to enter values in the order derived, but requires the values be specified, either by naming or by selection. Program-driven execution is best when the order is standard.)

- Is the type of execution appropriate to the type of interaction?
- If both types of execution are supported, is the type user-selectable or does the tool select the type based on the specific interaction?

Dynamic/Static Window

Does the tool/APSE allow presentations to be created, deleted, moved, or changed dynamically? (Dynamic windows are needed to display varying lists, objects of varying structure, recursive structures, script updates, and in-place expand and compress.)

Incremental/Batch Feedback

Does the tool/APSE report syntactic and semantic errors when they are made or in a batch at some later time or both? (Incremental feedback catches errors early, but may cause spurious error messages in the middle of a transaction. Batch feedback reduces the number of passes required of the tool before the operation can be successfully completed.)

- Is the type of feedback appropriate to the type of interaction?
- If both types of feedback are supported, is the type user-selectable or does the tool select the type based on the specific interaction?

Correcting Errors

Does the tool/APSE allow the user to undo any semantic actions caused by the entry of erroneous data?
Does the tool/APSE allow the user to correct syntactic, semantic, and user errors?
Does the tool/APSE allow for incremental error correction, i.e., correction should not require the reentry of correct data?
Does the tool/APSE allow syntactic and semantic errors to be corrected at any time, and user errors to be corrected until there has occurred some irreversible semantic action?

User Interface Paradigm

What is the user interface paradigm: form filling, text editing, teletype input/output, structure editing, or graphical display?

Figure 5.12-1 Analyzing User Interfaces Questionnaire (Continued)

5.13 ZEUS

Purpose: The hardware independence checkout software, Zeus, is a tool designed to parse C source, extract function calls, and through an expert systems rule base, compare those calls against standards (POSIX, System V Interface Definition, X-Open) and rules representing project restrictions, to enforce modularity and identify hardware dependent functions. **Note that use of this tool requires access to the C language source code.**

[@RM: Consistency	6.4.10;
@RM: Modularity	6.4.20;
@RM: Rehostability	6.4.26;
@RM: Software Production Vehicle(s)	6.4.31;
@RM: System Compatibility	6.4.35]

Primary References:

Host/OS: Unix

Vendors/Agents: [ACSI]

Method: Source code analysis and modification.

Inputs: Zeus, C source code for tool, and local compliance standards.

Process:
1. Obtain the Zeus tool.
2. Parse the C source code for the tool of interest.
3. Collect and analyze the results.
4. Optional - compile and link the modified source code.

Outputs: Problem report and modified source code.

6. COMPILATION SYSTEM ASSESSORS

For the purposes of this document, the compilation system is defined as those APSE components which are Ada-specific and are required for validation: the compiler, the code generator, the program library management system, and the runtime support system. While each of these components have characteristics which should be assessed individually, the assessment of their combined functionality will be more critical to the successful development of mission critical software.

The criticality of assessor development for these four compilation system components is made evident by the many large-scale projects with requirements for the use of Ada. These large-scale projects include the Strategic Defense Initiative (SDI), the NASA Space Station, the Software Technology for Adaptable, Reliable Systems (STARS) program, Army Tactical Command and Control System, Army WWMCCS Information System (WIS), and the Advanced Tactical Fighter (ATF), A-12, and Light Helicopter Experimental (LHX) programs being evaluated for common avionics systems under the auspices of the Joint Integrated Avionics Working Group (JIAWG). The successful performance of these systems depends upon the quality/extent of code generation support and execution support found in the compilation system.

6.1 Ada COMPILER VALIDATION CAPABILITY (ACVC)

Purpose: Validation of the completeness of the Ada compiler by means of a compiler test suite. The ACVC consists of a test suite, analysis tools, and accompanying documentation, to enable the determination of conformance of Ada compiler implementations to the ANSI/MIL-STD-1815A. Note: The AJPO requires that Ada compilers pass the ACVC and the vendor allow the distribution of the resulting Validation Summary Report (VSR) in order for the compiler to be advertised as a commercially available Ada compiler.

[(@RM: Compilation 7.1.6.7, @RM: Program Library Management 7.2.1.7), @RM: Completeness 6.4.9]

Primary References: [ACVC 1989] Ada Compiler Validation Procedures, Version 2.0, AJPO, May 1989.

Vendors/Agents: [AVF]

Method: Automated test suite.

Inputs: ACVC source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain latest ACVC test suite
2. Following documentation, compile and run tests
3. Use analysis tools on test outputs.

Outputs: Validation results; Validation Summary Report (VSR).

6.2 IDA BENCHMARKS

Purpose: Evaluation of the capacity and performance of the Ada compiler by means of a compiler test suite.

[@RM: Compilation	7.1.6.7,	(@RM: Capacity	6.4.6,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Storage Effectiveness	6.4.32);
@RM: Runtime Environment	7.2.3.5,	@RM: Processing Effectiveness	6.4.23]

Primary References: [Hook 1985] A.A. Hook, G.A. Riccardi, M. Vilot, and S. Welke, "User's Manual for the Prototype Ada Compiler Evaluation Capability (ACEC)," Version 1, Institute for Defense Analysis, IDA Paper P-1879, October 1985, DTIC Number AD A163 272.

Host/OS: VAX/VMS or any system that can read ANSI standard tapes.

Vendors/Agents: [SofTech OH]

Method: Test suite.

Inputs: IDA source code, Ada compiler and runtime system, and host (and target) computers.

Process:

1. Obtain test suite from agent
2. Compile and run Ada programs.

Outputs: Timing and storage measurements for individual language features.

6.3 Ada COMPILER EVALUATION CAPABILITY (ACEC)

Purpose: The Ada Compiler Evaluation Capability (ACEC) Version 2.0 was developed by Boeing Military Airplanes for the Ada Joint Program Office (AJPO) under the direction of the Air Force Wright Research and Development Center (WRDC). Its primary purpose is to provide the capability to determine the performance and usability characteristics of Ada compilation systems. The ACEC consists of the ACEC Software Product and three supporting documents: the ACEC User's Guide, the ACEC Reader's Guide, and the ACEC Version Description Document.

ACEC Software Product - The ACEC Software Product consists of performance tests, assessor tools, and support software. The software product makes it possible to:

- Compare the performance of several implementations
- Isolate the strong and weak points of a specific system, relative to other systems which have been tested
- Determine what significant changes were made between releases of a compilation system
- Predict performance of alternative coding styles
- Evaluate the clarity and accuracy of a system's diagnostic messages
- Determine whether the functional capabilities of a program library system are sufficient to accomplish a set of predefined scenarios
- Determine whether the functional capabilities of a symbolic debugger are sufficient to accomplish a set of predefined scenarios (see [7.11]).

The main emphasis of the ACEC is measuring execution time efficiency, but it also addresses code size efficiency and compile time efficiency. The test suite includes:

- Language feature tests
- Composite benchmarks
- Optimization tests
- Sorting programs
- Example avionics application.

The assessor tools provide assistance in evaluating symbolic debuggers, program library systems, and compiler diagnostics. The test suite does not include explicit tests for

existence of language features.

The support software is a set of tools and procedures which assist in preparing and executing the test suite, in extracting data from the results of executing the test suite, and in analyzing the performance measurements obtained. The support tools consist of:

- INCLUDE - assists in adapting programs to particular targets by performing source text inclusion
- FORMAT and MEL_DATA_CONSTRUCTOR - extract and format timing and code expansion data
- MEDIAN - compares results of performance tests of various systems
- SINGLE SYSTEM ANALYSIS - compares results of related tests from a single system

The ACEC Software Product is developed for a variety of targets and is distributed on one 9-track, 1600 bpi, VAX/VMS backup tape.

ACEC User's Guide - The ACEC User's Guide provides ACEC users with the information necessary to adapt and execute the ACEC Software Product. This guide explains how to use the support tools and how to deal with the problems which may occur in the process of adapting and executing the ACEC Software Product.

ACEC Reader's Guide - The ACEC Reader's Guide describes how users can interpret the results of executing the performance tests and assessor tools. This guide also covers the statistical significance of the numbers produced, the organization of the test suite, and the submission of error reports and change requests.

ACEC Version Description Document - The Version Description Document describes the ACEC Software Product as contained on the distribution tape. The document describes the compilation units, programs, test problems, and sample data contained on the distribution tape. This document contains a set of indexes which allow the user to identify each test, its primary purpose, its secondary and incidental purposes, related and comparison test problems, and applicable LRM section.

Document Distribution - The ACEC User's Guide, ACEC Reader's Guide, and the ACEC Version Description Document are available individually in hard copy form or as a package distributed on one 9-track, 1600 bpi, VAX/VMS backup tape. NOTE: The package is distributed as an ASCII text file embedded with LaTeX formatting commands.

ACEC Version 2.0 was released in May of 1990. ACEC Version 2.1, a maintenance release, is scheduled for March 1991.

[@RM: Compilation	7.1.6.7,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Operability	6.4.21,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Storage Effectiveness	6.4.32);
@RM: Program Library Management	7.2.1.7,	(@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Storage Effectiveness	6.4.32);
@RM: Runtime Environment	7.2.3.5,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Processing Effectiveness	6.4.23)]

Primary References: [ACEC 1990] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Reader's Guide," Air Force Wright Research and Development Center, D500-12471-1, May 1990.

[ACEC 1990a] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) User's Guide," Air Force Wright Research and Development Center, D500-12470-1, May 1990.

[ACEC 1990b] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Version Description Document," Air Force Wright Research and Development Center, D500-12472-1, May 1990.

Vendors/Agents: [DACS]

Method: Automated test suite and questionnaires.

Inputs: *ACEC source code, Ada compilation system and symbolic debugger, host and* target computer.

Process:

1. Obtain the ACEC.
2. Compile and run the tests according to the documentation.
3. Answer the questions by running the tools, reading the documentation, or asking the vendor.
4. Use the analysis tools on the test outputs.

Outputs: Reports and/or completed questionnaires containing the assessment of the tools for the selected tests. Raw data from individual tests can be aggregated together with MEDIAN to allow for gross comparisons among different compilers.

6.4 PIWG BENCHMARK TESTS

Purpose: Identification of performance characteristics of Ada compilers. A set of tests have been collected by the Performance Issues Working Group (PIWG) a volunteer subgroup of the Special Interest Group for Ada (SIGAda) of the Association for Computing Machinery (ACM). They have been in the public domain since 1986 and are updated periodically.

Section 7.2 of the cited reference contains a description of the PIWG test suite structure. It says that "... the current PIWG suite only addresses the time domain, that is, only run-time and compile-time measurements are made (CPU time as well as elapsed time). Of these, only the run-time tests are considered mature and truly useful at this time." The following tests or test groups are included:

- Clock resolution
- Dynamic storage allocation
- Language feature tests
  - C Tests -- task creation and termination overhead
  - D Tests -- dynamic elaboration overhead
  - E Tests -- exception related timing
  - F Tests -- Boolean flag overhead and coding style impact
  - G Tests -- Text_IO related timing
  - H Tests -- LRM Chapter 13 features
  - L Tests -- loop overhead
  - P Tests -- procedure call related timing
  - T Tests -- task related timing
  - Y Tests -- delay related.
- Runtime checks overhead
- Compilation speed and capacity tests
- Composite benchmarks (Hennessy, Whetstone, and Dhrystone)
- Application (Path tracking program with a Kalman filter)

[@RM: Compilation	7.1.6.7,	(@RM: Capacity	6.4.6,
		@RM: Processing Effectiveness	6.4.23);
@RM: Runtime Environment	7.2.3.5,	@RM: Processing Effectiveness	6.4.23]

Primary References: [PIWG 1990] "Ada Performance Issues," Ada Letters, Special Edition from SIGAda, the ACM Special Interest Group on Ada Performance Issues Working Group, Vol. X, Number 3, Winter 1990.

[Ada Performance Issues (PIWG Special Edition) 4.19]

Vendors/Agents: [PIWG]

Method: Automated test suite.

Inputs: PIWG source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain the latest PIWG tests (from PIWG Bulletin Board).
2. Compile and run tests according to the documentation.

Outputs: Reports on the compile and execution time of each test run.

6.5 UNIVERSITY OF MICHIGAN BENCHMARK TESTS

Purpose: Identification of the execution efficiency of the code generated by Ada compilers. The tests measure only the performance of isolated language features as they relate to real-time performance. This suite of tests was among the first to put Ada benchmarking on a sound scientific and theoretical basis. The suite forms the basis of much of the PIWG suite (see [6.4]), but does contain some tests not included in the PIWG suite, such as tests for the presence of garbage collection and manipulation of variables of type **time** and **duration**.

[@RM: Compilation 7.1.6.7, @RM: Processing Effectiveness 6.4.23]

Primary References: [Clapp 1986a] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, and T. Schultze, "Toward Real-Time Performance Benchmarks for Ada," Communications of the ACM, Volume 29, Number 8, August 1986, pp. 760-778.

[Clapp 1986] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, and T. Schultze, "Toward Real-Time Performance Benchmarks for Ada," Robotics Research Laboratory, Dept. of Electrical Engineering and Computer Science, The Univ. of Michigan, RSD-TR-12-86, July 1986.

Vendors/Agents: [UMich]

Method: Test suite.

Inputs: UMichigan source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain the UMichigan tests
2. Compile and run according to the documentation.

Outputs: Reports on the execution time of each test run.

6.6 MITRE BENCHMARK GENERATOR TOOL (BGT)

Purpose: Evaluation of the ability of an Ada compilation system to support development of very large systems in Ada. Under sponsorship of the Federal Aviation Administration, MITRE developed the Benchmark Generator Tool (BCT). The benchmark tests address capacity issues arising with large system developments. The initial version (1988) includes two types of tests: Library Capacity Tests and Dependency Maintenance Tests.

[@RM: Compilation	7.1.6.7,	@RM: Capacity	6.4.6;
@RM: Program Library Management	7.2.1.7,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Storage Effectiveness	6.4.32)]

Primary References: [Rainer 1986] S.R. Rainer, and T.P. Reagan, "User's Manual for the Ada Compilation Benchmark Generator Tool," MITRE Corporation, MTR-87W00192-01, January 1988.

Host/OS: Any for which an Ada compiler exists.

Vendors/Agents: [MITRE VA]

Method: Automated tool.

Inputs: BGT source code, Ada compiler, and host computer.

Process:

1. Obtain the BGT
2. Compile according to the documentation.

Outputs: Results of the above analysis, including capacity limits, link time, compilation time, etc.

6.7 UK Ada EVALUATION SYSTEM (AES)

Purpose: Evaluation of Ada compilers and associated linkers/loaders, program library systems, debuggers, and run-time libraries. A test suite and a methodology (AES) were developed by Software Sciences Ltd., under sponsorship of the UK Ministry of Defense (MoD). The British Standards Institute (BSI) has been sponsored by the MoD to provide an Ada Evaluation Service, using the AES. Interested parties, such as compiler vendors or potential compiler purchasers, may pay BSI to conduct an evaluation or to supply a copy of an existing evaluation report.

[@RM: Compilation	7.1.6.7,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Cost	6.4.11,
		@RM: Operability	6.4.21,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: Storage Effectiveness	6.4.32);
@RM: Program Library Management	7.2.1.7,	(@RM: Capacity	6.4.6,
		@RM: Processing Effectiveness	6.4.23);
@RM: Runtime Environment	7.2.3.5,	@RM: Processing Effectiveness	6.4.23]

Primary References: [Pierce 1986] R.H. Pierce, I. Marshall, and S.D. Blude, "An Introduction to the MoD Ada Evaluation System," Software Sciences Ltd., Report Number 5485, June 1986.

Host/OS: Any for which an Ada compiler exists.

Vendors/Agents: [BSI]

Method: Automated test suite and questionnaire.

Inputs: AES source code and questionnaire, Ada compiler and runtime system, and host (and target) computer.

Process: Pay BSI to do an evaluation or purchase an existing evaluation report.

Outputs: An Evaluation Report organized in a standard format.

6.8 COMPILATION CHECKLIST

Purpose: Evaluation of the power of compilation by developing a list of functional capabilities.

{@RM: Compilation 7.1.6.7, @RM: Power 6.4.22}

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capability checklist (see Table 6.8-1) and compiler documentation.

Process: Check off capabilities demonstrated during compiler runs or discussed in the documentation.

Outputs: A list of capabilities provided by the compiler.

Table 6.8-1 Compilation Capabilities Checklist

FEATURE	FOUND	NOTES
Conditional Compilation		
Incremental Compilation		
Debug Information Generation		
Enable/Disable Listing		
Errors Only Listing		
Error Identification		
Set Default Directory For Source		
Set Listing Width And Height		
Specify Different Program Library		
Specify Main Program		
Disable Use Of SYSTEM Library		
Change package SYSTEM		
Suppress All Run-Time Checks		
Compile Multiple Files		
Language Sensitive Editor Support		
Specify Error Limit		
Enable/Disable An Error Category		
Specify Optimization Parameters		
Syntax Only Checking		
Symbol Table		
Variable Set/Use Indications (Cross-reference)		
Object Code Listing		
Object Attribute Map		
Code Statistics		
Unidentified Compiler Options (Pragmas)		
Controlled Dynamic Storage		
Elaboration Control		
Inline Expansion Of Subprograms		
Interface With Other Languages		
Specify Memory Size		
Pack Data Representations In Memory		
Priority Control Of Concurrent Tasks		
Shared Variables		
Specify Storage Unit		
Specify Alternative System Characteristics		
Machine Code Mapping		
Machine Code Insertions		
Cross Compilation		
Error Reporting		
Exceptions List		
Identify Target Dependencies		
Save User Configuration		
Shared Generic Support		

6.9 PROGRAM LIBRARY MANAGEMENT CHECKLIST

Purpose: Evaluation of the completeness and power of program library management by developing a list of functional capabilities.

[@RM: Program Library Management 7.2.1.7, (@RM: Completeness 6.4.9, @RM: Granularity 6.4.17, @RM: Power 6.4.22)]

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.9-1) and program library manager documentation.

Process: Check off capabilities demonstrated by the program library manager or discussed in the documentation.

Outputs: A list of capabilities provided by the program library manager.

Table 6.9-1 Program Library Management Capabilities Checklist

FEATURE	FOUND	NOTES
Listing Information		
List Of Logical Unit Names		
Associated File Names For Unit		
Units Using Specified Unit		
Units Used By Specified Unit		
Size Information		
Time-Stamp Information		
Kind And Granularity Of Compilation Element		
Units Used To Construct Executable		
Completeness And Currency Check		
Automatic Build Capability		
Automatic Compilation/Recompilation		
Spawn Command Language Subprocess		
Create Structures		
Move Elements Between Libraries		
Move Elements Between Directories		
Remove Compilation Unit		
Obsolescence Management		
Library Access Control		
Read Only (Shared)		
Exclusive		

6.10 ARTEWG CATALOGUE OF Ada RUNTIME IMPLEMENTATION DEPENDENCIES

Purpose: The purpose of this document is best stated by the following quotation taken from the rationale section in the catalogue: "The main goal of this catalogue is to be the one place where all the areas of the Ada Reference Manual...which permit implementation flexibilities can be found." These implementation dependencies affect the performance and adaptation characteristics of the generated code. The text describes each known dependency by a number (which identifies the relevant section and paragraph in the Ada Reference Manual), a topic or title, a question which poses the implementation issue, a dependency type (either explicit or implicit), a rationale explaining why the dependency exists, and an Ada example to further clarify the dependency. (These descriptions could be used as the basis for an automated test suite.)

[@RM: Compilation	7.1.6.7,	(@RM: Anomaly Management	6.4.2,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Retargetability	6.4.28,
		@RM: Storage Effectiveness	6.4.32);
@RM: Runtime Environment	7.2.3.5,	(@RM: Anomaly Management	6.4.2,
		@RM: Storage Effectiveness	6.4.32)]

Primary References: [ARTEWG 1987] "Catalogue of Ada Runtime Implementation Dependencies," Association for Computing Machinery, Special Interest Group on Ada, Ada Runtime Environment Working Group, 1 December 1987.

Vendors/Agents: [ARTEWG]

Method: Questionnaire.

Inputs: Descriptions of implementation dependent features.

Process:

1. Select critical dependencies
2. Build and run tests for each dependency or ask vendor how dependencies are implemented
3. Select compiler and/or make coding standards based on results of step 2.

Outputs: Evidence showing how features are implemented.

6.11 ARTEWG RUNTIME ENVIRONMENT TAXONOMY

Purpose: Describes the basic elements of Ada runtime environments and provides a common vocabulary. The following excerpt is taken from the introduction to the Taxonomy section. "If a runtime environment for an Ada program is composed of a set of data structures, a set of conventions for the executable code, and a collection of predefined routines, then the question arises: what are examples of these elements, and moreover, what is the complete set from which such elements are taken when a particular runtime environment is built? ...It should be noted that the dividing line between the predefined runtime support library on one hand, and the conventions and data structures of a compiler on the other hand, is not always obvious. One Ada implementation may use a predefined routine to implement a particular language feature, while another implementation may realize the same feature through conventions for the executable code.... This taxonomy concerns itself primarily with those aspects of the runtime execution architecture which are embodied as routines in the runtime library. It does not treat issues of code and data conventions, nor issues related to particular hardware functionalities, in any great depth."

[@RM: Runtime Environment 7.2.3.5, @RM: Completeness 6.4.9]

Primary References: [ARTEWG 1988] "A Framework for Describing Ada Runtime Environments," Proposed by Ada Runtime Environment Working Group (SIGAda), Ada Letters, Volume VIII, Number 3, May/June 1988, pp. 51-68.

Vendors/Agents: [ARTEWG]

Method: Questionnaire.

Inputs: Questionnaire (see Table 6.11-1) and runtime environment documentation.

Process: Describe capabilities demonstrated by the runtime environment or discussed in the documentation.

Outputs: A description of capabilities provided by the runtime environment.

Table 6.11-1 Runtime Environment Taxonomy

CAPABILITY	DESCRIPTION
Runtime Execution Model Generated Code Number of Areas for Package Data Mechanism for Uplevel Referencing of Objects (static link or display) Subprogram Call Sequences Parameter Passing Mechanisms Register Usage Preservation of Registers across Subprogram Calls Representation of Pointers Implementation of Runtime Type Checks Data Structures in the RTE Division between Inline Code and Runtime Routines Tasking Constructs Memory Management Exception Management Attributes Miscellaneous Commonly Invoked Routines Use of Target Instruction Set Architecture User-Visible Interfaces to Extend the Runtime System	
Dynamic Memory Management Stack Management Heap Management Single Heap or One Heap/Task Arrangement of Storage for Collections Storage Reclamation None Explicit (Unchecked Deallocation) Garbage Collection Pragma Controlled	
Processor Management Block Tasks Unblock Tasks Selection of Tasks which Actually Run Task Priorities Assignment to Physical Processor	

Table 6.11-1 Runtime Environment Taxonomy (Continued)

CAPABILITY	DESCRIPTION
Interrupt Management Asynchronous Events Timer Interrupts I/O Interrupts Hardware Failures Others Program Synchronous Events Arithmetic Overflow Arithmetic Underflow Divide by Zero Others Address Clauses for Task Entries Interrupt Initialization Interrupt Resetting	
Time Management Package Calendar Delay Statement Implementation	
Exception Management Finds Exception Handler for Exception Transfers Control to Exception Handler	
Rendezvous Management	
Task Activation	
Task Termination Task Completion Task Termination Task Abortion	
I/O Management Predefined Packages from Chapter 14 of LRM Additional I/O Facilities	
Commonly Called Code Sequences Multi-Word Arithmetic Operations Block Moves String Operations Attribute Calculations Others	
Target Housekeeping Functions Starting the Execution Environment Determining the Hardware Environment Determining the Software Environment Processor Initialization Interrupt Initialization Other Terminating the Execution Environment	

6.12 COMPILER ASSESSMENT QUESTIONNAIRE

Purpose: The document presents a hierarchical breakdown of the compiler shown in Fig. 6.12-1. Requirements for each element in the hierarchy are listed for certain attributes. Each requirement is augmented by one or more questions which address the requirement.

[@RM: Compilation	7.1.6.7,	(@RM: Anomaly Management	6.4.2,
		@RM: Augmentability	6.4.4,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Consistency	6.4.10,
		@RM: Cost	6.4.11,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Granularity	6.4.17,
		@RM: Maturity	6.4.19,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Rehostability	6.4.26,
		@RM: Required Configuration	6.4.27,
		@RM: Retargetability	6.4.28,
		@RM: Storage Effectiveness	6.4.32,
		@RM: System Compatibility	6.4.35,
		@RM: Visibility	6.4.40);
@RM: Runtime Environment	7.2.3.5,	(@RM: Anomaly Management	6.4.2,
		@RM: Augmentability	6.4.4,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Consistency	6.4.10,
		@RM: Cost	6.4.11,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Granularity	6.4.17,
		@RM: Maturity	6.4.19,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Rehostability	6.4.26,
		@RM: Required Configuration	6.4.27,
		@RM: Retargetability	6.4.28,
		@RM: Storage Effectiveness	6.4.32,
		@RM: System Compatibility	6.4.35,
		@RM: Visibility	6.4.40)]

Primary References: [E&V Report 1984] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 1.0," 17 October 1984, Appendix B of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, November 1984, pp. B-45-B-85, DTIC Number AD A153 609.

[Requirements for E&V 4.5]

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Questionnaire and compiler documentation.

Process: Answer questions based on documentation, using the compiler, or asking the vendor.

Outputs: Completed questionnaire.

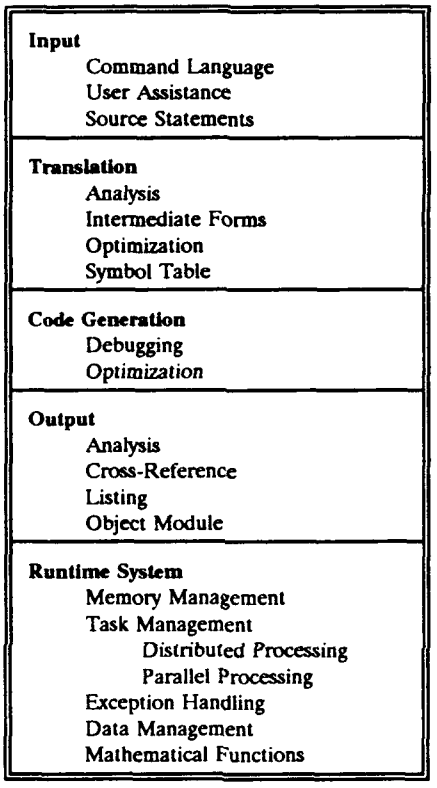

Figure 6.12-1 Compiler Hierarchy

6.13 WEIDERMAN: COMPILER EVALUATION LISTS

Purpose: This handbook "describes the dimensions along which a compilation system should be evaluated, enumerates some of the criteria that should be considered along each dimension, and provides guidance with respect to a strategy for evaluation." Table 6.13-1 below provides a "list of lists" found in Chapters 5, 6, and 8 of the handbook. Refer to the handbook itself for the actual elements of each list. In some cases the elements are simply listed; in other cases they are annotated with additional explanation and discussion.

[@RM: Compilation	7.1.6.7,	(@RM: Capacity	6.4.6,
		@RM: Cost	6.4.11,
		@RM: Maturity	6.4.19,
		@RM: Operability	6.4.21,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: Storage Effectiveness	6.4.32)]

Primary References: [Weiderman 1989] N.H. Weiderman, "Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0," Software Engineering Institute, Technical Report CMU/SEI-89-TR-13, March 1989, DTIC Number AD A207 717.

[Weiderman: Compiler Evaluation and Selection 4.18]

Vendors/Agents: [SEI]

Method: Checklists and Questionnaires

Inputs: Lists, characterization forms, and accompanying discussion (see Table 6.13-1).
Process: Employ lists to evaluate appropriate dimensions, as indicated in the handbook.
Outputs: Lists of capabilities and completed characterization forms.

Table 6.13-1 Compiler Evaluation Lists

Compile/Link-Time Issues (Chapter 5)

- Compiler options often provided (5.1.1)
- Implementation-defined pragmas (5.1.2)
- Other important compiler features (5.1.4)
- Factors influencing "lines of code per minute" (5.2)
- Questions to be answered (5.2)
- Capacities and limits tested by the AES [5.7] (5.2)
- Documentation characteristics (5.4.3)

Execution-Time Issues (Chapter 6)

- Features not likely to generate calls to the runtime system (6.2.7)
- Features likely to generate calls to the runtime system (6.7.2)
- Optimizations that can be performed (6.2.3)
- Features critical to tasking performance (6.4.1)
- Operations concerning exception handling whose overhead can be measured (6.4.2)
- Areas of concern related to space efficiency of the runtime system (6.5)
- Useful features of runtime systems (6.6)
- Questions related to interrupt handling (6.8)

Benchmark Issues (Chapter 8)

- Factors causing variation in results (8.2)
  - Memory effects
  - Processor effects
  - Operating and runtime system effects
  - Program translation effects
- Standard benchmark configuration information (8.7)
  - For the host system
  - For the target system
  - For all benchmarks

6.14 RUNTIME SUPPORT SYSTEM QUESTIONNAIRE

Purpose: Characterization and evaluation of the Runtime Support (RTS) system.

[@RM: Runtime Environment	7.2.3.5,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Distributedness	6.4.12,
		@RM: Functional Scope	6.4.15,
		@RM: Generality	6.4.16,
		@RM: Granularity	6.4.17,
		@RM: Modularity	6.4.20,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Reconfigurability	6.4.25,
		@RM: Retargetability	6.4.28,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35)]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 6.14-1) and runtime support system documentation.

Process: Answer questions based on documentation, using the runtime support system, or asking the vendor.

Outputs: Completed questionnaire.

Is support provided for	- single processor only? - multiple homogenous processors? - multiple heterogenous processors? • calls made to another process? • actual synchronization?
Is support provided for	- single programs? - multiple programs?
Is support provided for	- tight coupling (characterized by shared/common memory) - loose coupling (communicate but no shared/common memory)
Does the RTS	- require use of the operating systems? - accept instructions from the operation system? - replace the operating system?
Is the RTS	- modularly constructed? - modifiable (standard modifications or user-defined)? - sub-settable? - fault tolerant? - secure?
What are the language features that are supported? - How are they supported?	

Figure 6.14-1 Runtime Support System Questionnaire

6.15 HARTSTONE SYNTHETIC BENCHMARK

Purpose: The purpose of the first paper cited below "... is to define the operational concept for a series of benchmark requirements to be used to test the ability of a system to handle hard real-time applications. In the tradition of Whetstone and Dhrystone, we will call these the "Hartstone" benchmarks, where "Hart" is derived from Hard Real-Time. These will be synthetic benchmarks in the spirit of Whetstone and Dhrystone, and their definition will make it possible to compare a number of hardware/software/algorithm architectures. Several programs written in Ada for a specific machine configuration will be available from the SEI as examples of implementations of the requirements."

The following categories of tests are described:

- PH -- Periodic Tasks, Harmonic Frequencies
- PN -- Periodic Tasks, Non-Harmonic Frequencies
- AH -- PH Series with Aperiodic Processing Added
- SH -- PH Series with Synchronization
- SA -- PH Series with Aperiodic Processing and Synchronization

The second document cited below is a Hartstone User's Guide. It describes the structure and behavior of one category of Hartstone requirements, the Periodic Harmonic (PH) Test Series. It also provides guidelines for performing experiments and interpreting the results, as well as source code listings and information on how to obtain source code in machine-readable form.

[@RM: Compilation	7.1.6.7,	(@RM: Accuracy	6.4.1,
		@RM: Processing Effectiveness	6.4.23);
@RM: Runtime Environment	7.2.3.5,	(@RM: Accuracy	6.4.1,
		@RM: Processing Effectiveness	6.4.23)]

Primary References: [Weiderman 1989a] N.H. Weiderman, "Hartstone: Synthetic Benchmark Requirements for Hard Real-Time Applications," SEI-89-TR-23, Software Engineering Institute, June 1989. (reprinted in Ada Letters special PIWG issue [@PIWG 1990])

[Donohoe 1990] P. Donohoe, R. Shapiro, and N.H. Weiderman, "Hartstone Benchmark User's Guide, Version 1.0," SEI-90-UG-1, Software Engineering Institute, March 1990.

[Donohoe 1990a] P. Donohoe, R. Shapiro, and N.H. Weiderman, "Hartstone Benchmark Results and Analysis, Version 1.0," SEI-90-TR-7, Software Engineering Institute, June 1990.

[Ada Performance Issues (PIWG Special Edition) 4.19]

Vendors/Agents: [SEI]

Method: Test suite

Inputs: SEI source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain the Hartstone tests and documentation.
2. Modify the test code as necessary.
3. Compile and run according to the documentation.

Outputs: Reports on the number of missed deadlines for each test run.

6.16 Ada COMPILER PERFORMANCE TEST SUITE (ACPS)

Purpose: The purpose of the ACPS is to "assist users in evaluating the performance of runtime environments provided by Ada compilation systems." The test suite contains feature tests as well as composite tests. A unique feature of this suite is that there are Ada, JOVIAL, and FORTRAN files so that the three languages can be compared on machines that support these languages.

[@RM: Compilation	7.1.6.7,	(@RM: Processing Effectiveness	6.4.23,
		@RM: Storage Effectiveness	6.4.32);
@RM: Runtime Environment	7.2.3.5,	(@RM: Processing Effectiveness	6.4.23,
		@RM: Storage Effectiveness	6.4.32)]

Primary References: [Byrne 1990] D.J. Byrne and R.C. Ham, "Ada Versus FORTRAN Performance Analysis Using the ACPS," (pp. 139-145) of Ada Letters, Special Edition from SIGAda, the ACM Special Interest Group on Ada Performance Issues Working Group, Vol. X, Number 3, Winter 1990. [@PIWG 1990]

[Ada Performance Issues (PIWG Special Edition) 4.19]

Vendors/Agents: [Aerospace Corp.]

Method: Test suite.

Inputs: ACPS source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain the ACPS tests
2. Compile and run according to the documentation.

Outputs: Reports on the execution times and memory sizes of each test run.

6.17 PRODUCTION QUALITY Ada COMPILER (PQAC) TEST SUITE

Purpose: The Aerospace Corporation basically has taken the 149 requirements of "The Definition of a Production Quality Ada Compiler" (SD-TR-87-29) very literally and applied a test suite with a weighing scheme. It is mostly a pass-fail system of tests rather than a complete performance evaluation. The Definition has performance requirements, e.g., "The compiler shall compile a syntactically and semantically correct Ada program of at least 200 Ada source statements at a rate of at least 200 statements per minute (elapsed time), for each 1 MIPS of rated processing speed of the specified host computer, while meeting the object code requirements in 2.5.1 and 2.5.2." It has capacity requirements, e.g., 2,500,000 source statements in a program, 1024 type declarations, and 64 formal parameters in an accept. It has user interface requirements, e.g., "The compiler shall implement an option to disable the generation of diagnostic messages of a specified severity level." It has external tool interface requirements, e.g., "The compiler and/or linker/loader shall support the partial linking of object modules as specified by the user." It has Ada language requirements in the following categories: general, character sets, data representation, subprograms, tasking, exceptions, generics, interfaces with other languages, I/O, system information, and pragmas. It has quality assurance and reliability requirements, e.g., "Production quality compilers should exhibit an error rate of no more than 1 verified new error for each 250,000 new lines of Ada compiled. This rate shall decrease over time as the compiler matures." Finally there are documentation requirements, e.g., "The vendor shall provide a Run-time System Manual for each compiler configuration." There are Ada programs for all the requirements, even those that require inspection. It comes with a "code expander" (program generator) for the capacity tests.

The documents consist of two volumes. The first report outlines a procedure for using "The Definition of a Production Quality Ada Compiler" (SD-TR-87-29) as the basis for determining if an Ada compiler is of production quality. The report describes the development of a test suite from the requirements set forth in SD-TR-87-29, as well as the results of applying this test suite to two validated Ada compilers. An analysis of SD-TR-87-29 from creating and applying the test suite has also been provided (sic). The second volume contains the PQAC test suite source code and operating instructions. This test suite was derived from the requirements in the SD-TR-87-29.

[@RM: Compilation	7.1.6.7,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Maturity	6.4.19,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23)]

Primary References: [Petrick 1989] B.A. Petrick and S.J. Yanke, "An Analysis of *The Definition of a Production Quality Ada Compiler*," Engineering Group, The Aerospace Corporation, Volume I (SSD-TR-89-81) and Volume II (SSD-TR-89-82, PQAC Test Suite), 13 March 1989.

Vendors/Agents: [Aerospace]

Method: Test suite.

Inputs: PQAC source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain the PQAC tests
2. Compile and run according to the documentation.

Outputs: Raw data and pass/fail designation for each test run.

6.18 Ada COMPILER SPECIFICATION AND SELECTION QUESTIONNAIRES

Purpose: "The Ada language reference manual defines the language rather than indicating a list of the desirable properties of an implementation of the language. The purpose of this guide is to list the characteristics of an implementation that should be taken into account in the specification or selection of an Ada compiler."

[@RM: Compilation	7.1.6.7,	(@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Documentation Quality	6.4.13,
		@RM: Functional Overlap	6.4.14,
		@RM: Granularity	6.4.17,
		@RM: Modularity	6.4.20,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Rehostability	6.4.26,
		@RM: Required Configuration	6.4.27,
		@RM: Retargetability	6.4.28,
		@RM: Software Production Vehicle(s)	6.4.31,
		@RM: Storage Effectiveness	6.4.32,
		@RM: System Compatibility	6.4.35);
@RM: Runtime Environment	7.2.3.5,	(@RM: Processing Effectiveness	6.4.23,
		@RM: Storage Effectiveness	6.4.32)]

Primary References: [Nissen 1984] J.C.D. Nissen, B.A. Wichmann, et al., "Guidelines for Ada Compiler Specification and Selection," in *Ada: Language, Compilers and Bibliography*, ed. M.W.Rogers, Cambridge University Press, 1984.

[Nissen, et al: Guidelines for Ada Compiler Specification and Selection 4.17]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaires.

Inputs: Questionnaires (see Table 6.18-1) and compilation system documentation.

Process: Answer questions based on documentation, using the compilation system, or asking the vendor.

Outputs: Completed questionnaires.

Table 6.18-1 Ada Compiler Specification and Selection Questionnaires

ATTRIBUTE	RELEVANT SECTION(S) FROM NISSEN'S GUIDELINES
Capacity	5.1
Commonality	6.1
Documentation Quality	4.1-4.5, 6.
Functional Overlap	6.1
Granularity	6.1
Modularity	6.1
Operability	4.
Power	4.1-4.3, 4.6
Processing Effectiveness	
Compiler	5.
Runtime Environment	5.2, 6.2
Proprietary Rights	8.
Rehostability	2., 3., 7.3
Required Configuration	2., 7.3
Retargetability	2., 3., 7.2
Software Production Vehicle(s)	7.2
Storage Effectiveness	
Compiler	5.
Runtime Environment	5.2, 6.2
System Compatibility	6.

7. TARGET CODE GENERATION AIDS AND ANALYSIS TOOLSET ASSESSORS

These tools are used to assess host-target system cross-assemblers; host-based target linkers and loaders; host-based target system instruction-level simulators/emulators; host-based target-code symbolic debuggers; and host-based target system instrumentation interfaces which provide visibility into target processes during program execution. These assessments are also used in the case where the host computer is also the target computer.

7.1 ASSEMBLING CHECKLIST

Purpose: Evaluation of the power of assembling by developing a list of functional capabilities.

[@RM: Assembling 7.1.6.6, @RM: Power 6.4.22]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 7.1-1) and assembler documentation.

Process: Check off capabilities demonstrated during assembler runs or discussed in the documentation.

Outputs: A list of capabilities provided by the assembler.

Table 7.1-1 Assembling Capabilities Checklist

FEATURE	FOUND	NOTES
Code Generation		
Macro Preprocessing		
Conditional Assembly		
Debug Information Generation		
Enable/Disable Listing		
Errors Only Listing		
Set Listing Width and Height		
Suppress All Run-Time Checks		
Assemble Multiple Files		
Specify Error Limit		
Enable/Disable An Error Category		
Syntax Only Checking		
Symbol Table		
Code Statistics		
Cross Assembly		

7.2 LINKING/LOADING CHECKLIST

Purpose: Evaluation of the power of linking/loading by developing a list of functional capabilities.

[@RM: Linking/Loading 7.1.6.13, @RM: Power 6.4.22]

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 7.2-1) and linker/loader documentation.

Process: Check off capabilities demonstrated during linker/loader runs or discussed in the documentation.

Outputs: A list of capabilities provided by the linker/loader.

Table 7.2-1 Linking/Loading Capabilities Checklist

FEATURE	FOUND	NOTES
Non-Specific Language Linking		
Deferred (After A Specific Time)		
Enable/Disable Link Map Generation		
Specify Full/Brief Link Map		
Generate A Link Command File		
Enable/Disable Symbol Cross-Reference		
Generate Debug Information		
Enable/Disable Execution File Creation		
Specify Batch/Nobatch Operation		
Specify Map File Name		
Specify Object File Name		
Specify Diagnostic Output File		
Enable/Disable System Library Search		
Enable/Disable Traceback Information		
Library Search Capabilities		
Extended Options Capabilities		
Sharable Image Support		
Specify Maximum Memory		
Specify Optimization Parameters		
Force Load		
Enable/Disable Library Trace		
Specify Main Program		
Non-Specific Language Main Program		
Overlays		
Link-Time Dead Code Elimination		

7.3 IMPORT/EXPORT CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of import/export by developing a list of functional capabilities.

[@RM: Import/Export 7.2.3.6, @RM: Completeness 6.4.9]

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 7.3-1) and import/export documentation.

Process: Check off capabilities demonstrated by the import/export system or discussed in the documentation.

Outputs: A list of capabilities provided by the import/export system.

Table 7.3-1 Import/Export Capabilities Checklist

FEATURE	FOUND	NOTES
Host to Target Object Downloading Target to Host Data Uploading		

Note: This table will be expanded in a future version of the Guidebook.

7.4 EMULATION CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of emulation by developing a list of functional capabilities.

[@RM: Emulation 7.3.2.13, @RM: Power 6.4.22]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 7.4-1) and emulation system documentation.

Process: Check off capabilities demonstrated by the emulation system or discussed in the documentation.

Outputs: A list of capabilities provided by the emulation system.

Table 7.4-1 Emulation Capabilities Checklist

FEATURE	FOUND	NOTES
Session security (lock-out unauthorized users)		
RS-232 interface to host (portable among hosts)		
Replaceable target pods (portable among targets)		
Support for simulating hardware devices		
Switching screen (user vs. debug displays)		
Read/write access to program library symbols		
Runtime controls of the state of the emulator		
Read/write access to target memory and I/O		
Full-speed execution with active breakpoints		
Full-speed execution while tracing		
Dynamic window for variables		
Multi-task tracing		
Exception tracing		

7.5 DEBUGGING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness and power of debugging by developing a list of functional capabilities.

[@RM: Debugging 7.3.2.5, (@RM: Completeness 6.4.9, @RM: Power 6.4.22)]

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 7.5-1) and debugger documentation.

Process: Check off capabilities demonstrated by the debugger or discussed in the documentation.

Outputs: A list of capabilities provided by the debugger.

Table 7.5-1 Debugging Capabilities Checklist

FEATURE	FOUND	NOTES
Instrumentation		
Statement		
Branch		
Block		
CSU		
CSC		
Machine Level Debugging		
Host-Based Target Debugging		
Support for Debugging Multiple Processors from Single Terminal		
Customization of Debugger for New Target Environment		
Symbolic Debugging		
Tracing		
Breakpoint Control		
Data Flow Tracing		
Path Flow Tracing		
Selectable Level Of Granularity		
Display		
Program Source		
History		
Stack (Calling Hierarchy)		
Tasks		
Rendezvous Status		
Breakpoints		
Tracepoints		
Memory		
Collections And Global Heaps		
Name Of Current Exception		
Evaluate Objects		
Step		
Single		
By Discrete Amounts		
Into Subprograms		
Over Subprograms		
To Next Scheduling Event		
To Next Exception		
To End of Program Unit		
Miscellaneous		
Symbol Abbreviation		
Set Context For Control		
Input Debugger Command Files		
Modify Variable Values		
Modify Object Code		
Modify Control Flow		
Console Interrupt		
Full Screen Mode		
Keypad For Entering Commands		
Virtual Clock		
Special Compilation Mode		
Multi-Language Support		
Complete Ada Language Support with Deep Nesting		
Dynamic Interrupt		
Optimization Support		
Units Comprising Executable		
Locate Objects with Overloaded Names		
No Overhead to Explicitly Create a Debug File		

7.6 TIMING ANALYSIS CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of timing analysis by developing a list of functional capabilities.

[@RM: Timing Analysis 7.3.2.14, @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 7.6-1) and timing analysis system documentation.

Process: Check off capabilities demonstrated by the timing analysis system or discussed in the documentation.

Outputs: A list of capabilities provided by the timing analysis system.

Table 7.6-1 Timing Analysis Capabilities Checklist

FEATURE	FOUND	NOTES
Timing Instrumentation Intrusive Non-Intrusive User Specified Error Tolerances Use of Timing Loop Repetitive Execution Until Stable Convergence Measurement of Overhead Execution Test for Clock Jitter System Clock Accuracy Consideration Hardware Organization (Cache, Pipeline ...) Considerations Operating System (Virtual, Multiprocessing...) Considerations Size of Test Problem Considerations		
Fraction By Section Of Code		
Tasking Monitor Fraction Executing Fraction Runnable Fraction Runnable and not Executing Time Between Runnable and Executing Time Between Events System Idle Time		
Miscellaneous Timing Loop Code is System Independent Special Hardware is not Needed Code to be Measured is Easily Installed Output Shows Variations in Measurements Statistical Measurements are Available Use of Computer Resources is Minimized Measures Either Wall or Clock Time		

7.7 REAL-TIME ANALYSIS CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of real-time analysis by developing a list of functional capabilities.

[@RM: Real-Time Analysis 7.3.2.17, @RM: Completeness 6.4.9]

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 7.7-1) and real-time analysis system documentation.

Process: Check off capabilities demonstrated by the real-time analysis system or discussed in the documentation.

Outputs: A list of capabilities provided by the real-time analysis system.

Table 7.7-1 Real-time Analysis Capabilities Checklist

FEATURE	FOUND	NOTES
Hardware-In-The-Loop Non-Intrusive Instrumentation Performance Analysis Symbolic Trace		

7.8 INSTRUCTION-LEVEL SIMULATION CHECKLIST

Purpose: Evaluation of the completeness of instruction-level simulation by developing a list of functional capabilities.

[@RM: Simulation and Modeling 7.3.2.3, @RM: Completeness 6.4.9]

Primary References: [Weiderman 1987a] N.H. Weiderman, et al., "Ada for Embedded Systems: Issues and Questions," Software Engineering Institute, Technical Report CMU/SEI-87-TR-26, December 1987, DTIC Number AD A191 096.

Vendors/Agents: [SEI]

Method: Checklist.

Inputs: Capabilities checklist (see Table 7.8-1) and instruction-level simulation system documentation.

Process: Check off capabilities demonstrated by the instruction-level simulation system or discussed in the documentation.

Outputs: A list of capabilities provided by the instruction-level simulation system.

Table 7.8-1 Instruction-level Simulation Checklist

FEATURE	FOUND	NOTES
Accurately simulates both the functional and temporal behavior of the target's instruction set architecture		
Provides access to all memory locations and registers		
Supports typical features found in a symbolic debugger Single-step instruction execution Examines variable values Start/stop program execution		
Performs timing analysis Provides assembler instruction execution times Provides Ada instruction execution times Provides Ada subprogram execution times		
Supports simulated input/output interaction Provides access to I/O ports Provides access to device control and data registers Emulates the architecture's interrupt mechanism		
Facilitates the set-up and reuse of test sessions Freezes the current session's context Executes debugger commands from script files Supplies I/O data from existing data files		

7.9 SEI DEBUGGING EXPERIMENT

Purpose: Evaluation of an environment's capabilities, from the point of view of the unit tester. An experiment was designed to simulate the activities normally associated with small projects, namely the design, creation, modification, and testing of a single unit or module. See also the SEI Design Support Experiment [10.1].

[@RM: Debugging 7.3.2.5, (@RM: Power 6.4.22, @RM: Processing Effectiveness 6.4.23)]

Primary References: [Weiderman 1987] N.H. Weiderman and A.N. Habermann, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, Chapter 5, DTIC Number AD A180 905.

[Weiderman: Evaluation of Ada Environments 4.13]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment.

Inputs: The "generic" experiment description, an APSE, and host (and target) computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in table of functional elements present and missing, elapsed time and cpu time values, and subjective judgments based on the experience.

7.10 Ada-EUROPE: DEBUGGING QUESTIONNAIRE

Purpose: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@Buxton 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided. Chapter 15 discusses issues associated with testing, debugging, and dynamic analysis (coverage analysis, symbolic evaluation, mutation analysis, facilities for symbolic interaction, breakpoints and single stepping, tracing, multitasking and real time features, debugging of multiprocessor systems, monitor on the target, simulation, emulation, and other tools for dynamic analysis).

[@RM: Debugging 7.3.2.5, @RM: Completeness 6.4.9]

Primary References: [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986, Chapter 15.

[Ada-Europe: Selecting an Ada Environment 4.9]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire.

Inputs: Questionnaire, tool, and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed questionnaire.

7.11 ACEC SYMBOLIC DEBUGGER QUESTIONNAIRE

Purpose: The Ada Compiler Evaluation Capability (ACEC) Version 2.0 was developed by Boeing Military Airplanes for the Air Force Wright Research and Development Center (WRDC). Its primary purpose is to provide the capability to determine the performance and usability characteristics of Ada compilation systems. The ACEC consists of the ACEC Software Product and three supporting documents: the ACEC User's Guide, the ACEC Reader's Guide, and the ACEC Version Description Document.

ACEC Software Product - The ACEC Software Product consists of performance tests, assessor tools, and support software. Among other things (see [6.3]), the software product makes it possible to determine whether the functional capabilities of a symbolic debugger are sufficient to accomplish a set of predefined scenarios. The main emphasis of the ACEC is measuring execution time efficiency, but it also addresses code size efficiency and compile time efficiency. The assessor tools provide assistance in evaluating symbolic debuggers, program library systems, and compiler diagnostics. The support software is a set of tools and procedures which assist in preparing and executing the test suite, in extracting data from the results of executing the test suite, in analyzing the performance measurements obtained.

The ACEC Software Product is developed for a variety of targets and is distributed on one 9-track, 1600 bpi, VAX/VMS backup tape.

ACEC User's Guide, ACEC Reader's Guide, ACEC Version Description Document - For descriptions, see [6.3].

Document Distribution - The ACEC User's Guide, ACEC Reader's Guide, and the ACEC Version Description Document are available individually in hard copy form or as a package distributed on one 9-track, 1600 bpi, VAX/VMS backup tape. NOTE: The package is distributed as an ASCII text file embedded with LaTeX formatting commands.

The second version of the ACEC was released in May 1990.

[@RM: Debugging	7.3.2.5,	(@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22)]

Primary References: [ACEC 1990] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Reader's Guide," Air Force Wright Research and Development Center, D500-12471-1, May 1990.

[ACEC 1990a] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) User's Guide," Air Force Wright Research and Development Center, D500-12470-1, May 1990.

[ACEC 1990b] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Version Description Document," Air Force Wright Research and Development Center, D500-12472-1, May 1990.

Vendors/Agents: [DACS]

Method: Automated test suite and questionnaires.

Inputs: ACEC source code, Ada compilation system and symbolic debugger, host and target computer.

Process:

1. Obtain the ACEC.
2. Compile and run the tests according to the documentation.
3. Answer the questions by running the tools, reading the documentation, or asking the vendor.
4. Use the analysis tools on the test outputs.

Outputs: Reports and/or completed questionnaires containing the assessment of the tools for the selected tests.

8. TEST SYSTEMS ASSESSORS

These assessors examine the ability of the APSE or APSE component to support and facilitate the planning, development, execution, evaluation, and documentation of tests of software.

8.1 TESTING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness and power of testing by developing a list of functional capabilities.

[@RM: Analysis 7.3, (@RM: Completeness 6.4.9, @RM: Power 6.4.22)]

Primary References: [DeMillo 1986] R.A. DeMillo, "Functional Capabilities of a Test and Evaluation Subenvironment in an Advanced Software Engineering Environment," Georgia Institute of Technology GIT-SERC-86/07, 20 October 1986.

Vendors/Agents: [GIT]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 8.1-1) and testing system documentation.

Process: Check off capabilities demonstrated by the testing system of discussed in the documentation.

Outputs: A list of capabilities provided by the testing system.

Table 8.1-1 Testing Capabilities Checklist

FEATURE	FOUND	NOTES
Static Analyzers Code Auditors Consistency Checkers Interface Analyzers Completeness Checkers		
Tool Building Services Common "Front-End" Facilities for Languages of Interest (Parsing, Source & Internal Form Manipulation, Execution Facilities) Tool Composition Aids		
Test Building Services (including Test Data Generators) Symbolic Evaluators Component Coverage Analyzers Data Flow Analyzers Assertion Processors Mutation Analyzers Path and Domain Selection Aids Random Test Generators		
Test Description and Preparation Services Data Editors Data Auditors Body/Stub Generators File Comparators Data/File Services Software and System Test Communications Facilities		
Test Execution Services Test Harness Generator Data and Error Logging Services Quality Measurement Tools		
Test Analysis Services Correctness Analyzers (Oracles) Instrumentation Aids Status Display Tools Data Reduction and Analysis Tools Cross-reference (Traceability) Management and Analysis Tools		
Decision Support Services Documentation Services Information Repositories Problem Report Processing and Analysis Tools Change Request Processing and Analysis Tools		

8.2 SEI UNIT TESTING EXPERIMENT

Purpose: Evaluation of an environment's capabilities, from the point of view of the unit tester. An experiment was designed to simulate the activities normally associated with small projects, namely the design, creation, modification, and testing of a single unit or module. See also the SEI Design Support Experiment [10.1].

[@RM: Dynamic Analysis 7.3.2, (@RM: Power 6.4.22, @RM: Processing Effectiveness 6.4.23)]

Primary References: [Weiderman 1987] N.H. Weiderman and A.N. Habermann, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, Chapter 5, DTIC Number AD A180 905.

[Weiderman: Evaluation of Ada Environments 4.13]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment.

Inputs: The "generic" experiment description, an APSE, and host (and target) computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in table of functional elements present and missing, elapsed time and cpu time values, and subjective judgments based on the experience.

8.3 STEM/SAIC TEST TOOLS EVALUATION

Purpose: Evaluation of the capabilities of test tools. The goals of this project are to develop evaluation procedures, identify and classify test tools, evaluate some tools, and contribute to the STEM database [@STSC 1990]. A hierarchical checklist has been developed, including the following top-level categories:

Tool Information
Functions
Configuration Requirements
User Information
Vendor
Modification and Maintenance
Reliability
Contractual Matters
Cost
Training
Documentation
Installation
Operation

The cited reference provides second and third level entries under the above categories.

[@RM: Comparison	7.3.1.1,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Augmentability	6.4.4,
		@RM: Autonomy	6.4.5,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Consistency	6.4.10,
		@RM: Cost	6.4.11,
		@RM: Distributedness	6.4.12,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Maturity	6.4.19,
		@RM: Modularity	6.4.20,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Reconfigurability	6.4.25,
		@RM: Rehostability	6.4.26,
		@RM: Required Configuration	6.4.27,
		@RM: Retargetability	6.4.28,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: Simplicity	6.4.30,
		@RM: Storage Effectiveness	6.4.32,
		@RM: Traceability	6.4.36,
		@RM: Training	6.4.37,
		@RM: Vendor Support	6.4.38,
		@RM: Visibility	6.4.40);
@RM: Auditing	7.3.1.22,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Augmentability	6.4.4,
		@RM: Autonomy	6.4.5,

		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Consistency	6.4.10,
		@RM: Cost	6.4.11,
		@RM: Distributedness	6.4.12,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Maturity	6.4.19,
		@RM: Modularity	6.4.20,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Reconfigurability	6.4.25,
		@RM: Rehostability	6.4.26,
		@RM: Required Configuration	6.4.27,
		@RM: Retargetability	6.4.28,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: Simplicity	6.4.30,
		@RM: Storage Effectiveness	6.4.32,
		@RM: Traceability	6.4.36,
		@RM: Training	6.4.37,
		@RM: Vendor Support	6.4.38,
		@RM: Visibility	6.4.40);
@RM: Dynamic Analysis	7.3.2,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Augmentability	6.4.4,
		@RM: Autonomy	6.4.5,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Consistency	6.4.10,
		@RM: Cost	6.4.11,
		@RM: Distributedness	6.4.12,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Maturity	6.4.19,
		@RM: Modularity	6.4.20,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Reconfigurability	6.4.25,
		@RM: Rehostability	6.4.26,
		@RM: Required Configuration	6.4.27,
		@RM: Retargetability	6.4.28,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: Simplicity	6.4.30,
		@RM: Storage Effectiveness	6.4.32,
		@RM: Traceability	6.4.36,
		@RM: Training	6.4.37,
		@RM: Vendor Support	6.4.38,
		@RM: Visibility	6.4.40)]

Primary References: [Hampton 1990] J.Hampton, D.Dyer, and G.Daich, "STEM for Test Tools," USAF STSC - HQ USAF/SC Joint Software Conference Proceedings, Salt Lake City, 23-26 April 1990 [@STSC 1990].

[Software Tool Evaluation Model (STEM) 4.20]

Vendors/Agents: [STSC, SAIC]

Method: Capabilities Checklist and Questionnaire

Inputs: Capabilities checklist, questionnaire, test tool, and its documentation.

Process: 1) Check off capabilities demonstrated by testing or discussed in the documentation.

2) Record information about the test tool requested by the questionnaire.

Outputs: A list of capabilities provided by the tool and other important information about the tool.

8.4 Ada-EUROPE: TESTING AND DYNAMIC ANALYSIS QUESTIONNAIRE

Purpose: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@Buxton 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided. Chapter 15 discusses issues associated with testing, debugging, and dynamic analysis (coverage analysis, symbolic evaluation, mutation analysis, facilities for symbolic interaction, breakpoints and single stepping, tracing, multitasking and real time features, debugging of multiprocessor systems, monitor on the target, simulation, emulation, and other tools for dynamic analysis).

[(@RM: Simulation and Modeling	7.3.2.3,		
@RM: Coverage/Frequency Analysis	7.3.2.8,		
@RM: Mutation Analysis	7.3.2.9,		
@RM: Symbolic Execution	7.3.2.10,		
@RM: Resource Utilization	7.3.2.12,		
@RM: Emulation	7.3.2.13,		
@RM: Real-Time Analysis	7.3.2.17),	@RM: Completeness	6.4.9]

Primary References: [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986, Chapter 15.

[Ada-Europe: Selecting an Ada Environment 4.9]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire.

Inputs: Questionnaire, tool, and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed questionnaire.

9. TOOL SUPPORT COMPONENT ASSESSORS

These assessors evaluate or validate implementations of specifications for tool support components. Components that may be assessed could include a CAIS or a CAIS-A implementation, Portable Common Tool Environment (PCTE) implementations, and Ada language interfaces to the UNIX operating system and its variants (e.g., Berkeley UNIX, System V, A/UX, POSIX). Also included here are window managers (such as X-windows), language bindings to standard interface specification implementations (such as Ada bindings to GKS or SQL), I/O pipes, and RAM cache.

9.1 CAIS IMPLEMENTATION VALIDATION CAPABILITY (CIVC)

Purpose: The CIVC contract, awarded to SofTech, Inc., is developing a validation test suite to assess conformance of DoD-STD-1838 (CAIS) and MIL-STD-1838A (CAIS-A) implementations. The CAIS (Common APSE Interface Set) is an extensive set of interfaces to support the development of transportable tools for use in Ada Programming Support Environments (APSEs). The CIVC has successfully applied information modeling to the test coverage design and assessment required for validation testing. Hypermedia has been used for the delivery of test requirements, test designs and their associated traceability relationships. Version 1 of the CIVC (CIVC1) was delivered in February 1990 and comprises a test suite of over 250 tests. SofTech is preparing Version 2 of the CIVC (CIVC2) for delivery in February 1992 and will include over 500 tests. A beta version of CIVC2 derived from the porting of CIVC1 and containing 147 tests was delivered in July 1990. CIVC1 assesses conformance to DoD-STD-1838, while CIVC2 assesses conformance to MIL-STD-1838A. The CIVC is funded through the Ada Joint Program Office.

The major software components of the CIVC are the Framework, the Test Administrator, and the Test Suite. The documentation components consist of the Implementor's Guide, the Framework, the Test Report Reader's Guide, and the Operator's Guide. Each component is briefly described below.

CIVC Framework - The CIVC Framework is a hypertext-based product that provides complete and unique traceability between DoD-STD-1838 and the CIVC1 software product and MIL-STD-1838A and the CIVC2 software product. This framework product provides the means for evaluating the correctness of the CIVC product by determining the completeness and consistency through the development of a full traceability framework. CIVC has developed hypermedia systems for active presentation of test suite traceability. These systems significantly increase users' understanding of the derivation of software products.

CIVC Test Administrator - The Test Administrator provides the CIVC user interfaces encapsulating any target environment dependencies for operating the CIVC, and schedules and executes the CIVC validation tests defined in the Test Suite.

CIVC Test Suite - Test objectives, scenarios, test cases, static compilation tests, and a report manager constitute the CIVC Test Suite. The Test Suite encapsulates the actual tests which exercise a CAIS implementation and is organized by superclasses. Superclasses are arbitrary organizations of test classes. Test classes are groups of test cases which either: 1) have similar preconditions, or 2) have preconditions which depend on the postcondition of a previously executed test case. A preliminary validation capability of MIL-STD-1838A was delivered in July 1990 as the CIVC2 Beta Test Suite. Test selection is accomplished by automated analysis and prioritization systems developed by SofTech. The systems use information models to intelligently select the tests to be developed for the CIVC. Coverage of the test suite is increased by these methods, and the development effort consequently is more cost effective.

CIVC Implementor's Guide - The CIVC Implementor's Guide presents the conformance

requirements (test objectives) for a CAIS implementation, as well as the top level designs (scenarios) for the test cases that will validate a CAIS implementation's conformance to DoD-STD-1838 and MIL-STD-1838A. Also included are the conventions and rationale used in the development of the CIVC products.

CIVC Framework - See the discussion under the software products above.

CIVC Test Report Reader's Guide - This guide describes the format of the CIVC Conformance Report and how to interpret the data presented in the Conformance Report.

CIVC Operator's Guide - The Operator's Guide details the host system requirements, installation, and operation of the CIVC.

Collectively, these products: 1) discover and report ambiguities, incomplete parts, and other potential impediments to common interpretations of CAIS/CAIS-A, and 2) produce mechanisms for analyzing and reporting errors in CAIS/CAIS-A implementations that violate specifications. The beta release of the CIVC2 provides early validation support to CAIS-A implementors.

[@RM: Kernel 7.2.3.3, @RM: Completeness 6.4.9]

Primary References: [CIVC 1989] "CIVC Implementor's Guide," CIVC-FINL-19-1, SofTech, Inc., 16 October 1989.

[CIVC 1990] "CIVC1 Framework," CVC-VREL-2/1-01, SofTech, Inc., 1 March 1990.

[CIVC 1990a] "Test Report Reader's Guide with Appendix 1 - Operator's Guide," CIVC-FINL-020-02, SofTech, Inc., 19 March 1990.

[CIVC 1990b] "CIVC 2.0 Beta Test Suite Operator's Guide," CIVC-FINL-20-03, 18 July 1990.

[CAIS and CAIS-A: DoD-STD-1838 and MIL-STD-1838A 4.16]

Vendors/Agents: [AJPO, SofTech TX]

Method: Automated test suite.

Inputs: The CIVC test suite, CAIS implementation, Ada compiler and runtime system, and host computer.

Process:
1. Obtain the CIVC test suite.
2. Compile and run the tests.
3. Collect and analyze the results.

Outputs: Report describing the conformance of various aspects of the CAIS implementation to DoD-STD-1838/MIL-STD-1838A.

9.2 TOOL SUPPORT INTERFACE EVALUATION

Purpose: Evaluation of tool support interfaces in terms of four criteria: level, appropriateness, implementability, and performance. Five "scenarios" were designed and used to exercise a prototype CAIS implementation and a prototype PCTE implementation. The scenarios involved a configuration management system, an edit-compile-link-test cycle, a conference management system, a window manager, and a design editor.

[@RM: Kernel	7.2.3.3,	(@RM: Granularity	6.4.17,
		@RM: Operability	6.4.21,
		@RM: Processing Effectiveness	6.4.23,
		@RM: System Clarity	6.4.34)]

Primary References: [Long 1988] F.W. Long, and M.D. Tedd, "Evaluating Tool Support Interfaces," *Ada in Industry*, Proceedings of the Ada-Europe Conference, Munich, 7-9 June 1988, Cambridge University Press, 1988.

Host/OS: Sun

Vendors/Agents: [Cambridge University Press, College of Wales]

Method: Structured experiment

Inputs: The source code for the scenarios, the tool support interface(s) (CAIS, PCTE, other), Ada compiler and runtime system, and host computer.

Process:
1. Obtain the source code for the scenarios
2. Compile and run the scenario(s)
3. Collect the results.

Outputs: Objective results and subjective conclusions concerning the impact on tool writers and the cost and behavior of the interface implementation.

9.3 COMMAND LANGUAGE INTERPRETER ASSESSMENT QUESTIONNAIRE

Purpose: The document presents a hierarchical breakdown of the command language interpreter shown in Fig. 9.3-1. Requirements for each element in the hierarchy are listed for addressing certain attributes. Each requirement is augmented by one or more questions which address the requirement.

[@RM: Command Language Processing	7.2.3.1,	(@RM: Anomaly Management	6.4.2,
		@RM: Augmentability	6.4.4,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Consistency	6.4.10,
		@RM: Cost	6.4.11,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Granularity	6.4.17,
		@RM: Maturity	6.4.19,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Rehostability	6.4.26,
		@RM: Required Configuration	6.4.27,
		@RM: Storage Effectiveness	6.4.32,
		@RM: System Compatibility	6.4.35,
		@RM: Visibility	6.4.40)]

Primary References: [E&V Report 1984] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 1.0," 17 October 1984, Appendix B of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, November 1984, pp. B-39 - B-44, DTIC Number AD A153 609.

[Requirements for E&V 4.5]

Vendors/Agents: [E&V Team]

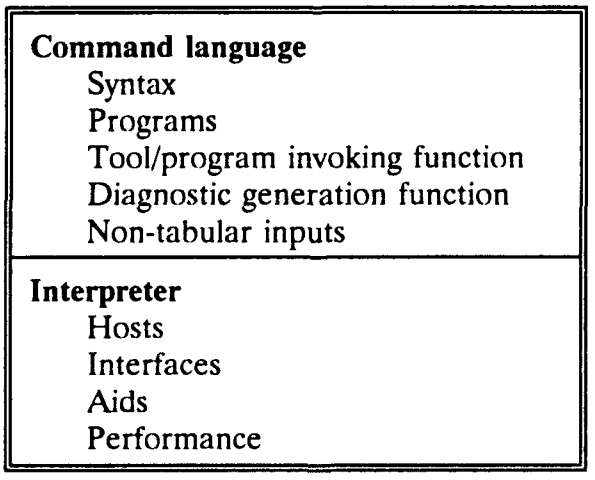

Figure 9.3-1 Command Language Interpreter Hierarchy

Method: Questionnaire.
Inputs: Questionnaire, command language interpreter, and documentation.
Process: Answer the questions by using the command language interpreter, reading the documentation, or asking the vendor of the command language interpreter.
Outputs: Completed questionnaire.

9.4 Ada-EUROPE: META-TOOLS AND TOOL COMPONENTS QUESTIONNAIRE

Purpose: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@Buxton 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided. Chapter 9 discusses issues associated with meta-tools and tool components (analyzer and editor generators, user interface support, data access generators, modifying tools and extending toolsets, utility packages, and applications generators).

[@RM: Program Generation	7.1.7.3,	(@RM: Completeness @RM: Power	6.4.9, 6.4.22)]

Primary References: [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986, Chapter 9.

[Ada-Europe: Selecting an Ada Environment 4.9]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire.
Inputs: Questionnaire, tool, and tool documentation.
Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.
Outputs: Completed questionnaire.

9.5 AIM BENCHMARK SUITES

Purpose: AIM Technology has developed several benchmark test suites to measure the performance of workstations to determine their appropriateness for various tasks or applications primarily for Unix systems. The suites examine workstation performance, multi-user performance, Unix utilities performance, X Windows performance, and Unix subsystem performance.

The **Suite V Workstation Benchmarks** test Unix and OS/2 single user, multi-tasking workstation environments. The benchmark tools stress the system by increasing the consumption of resources within an application and measuring the effect. To model user behavior, the benchmarks behave as a real user. The products load a diverse set of applications; timing and testing while moving from one application to another at odd intervals. The Workstation Benchmarks use 40 tests to evaluate system performance under user-defined application mixes such as: accounting, scientific operations, software development, spreadsheet, and database management. Functional tests can also be used to evaluate system performance under user-defined application mixes. Functional elements include: disk, math, floating point, inter-process communication, logic, and memory.

[(@RM: Database Management	7.2.1.1,		
@RM: Input/Output Support	7.2.3.2,		
@RM: Kernel	7.2.3.3,		
@RM: Math/Statistics	7.2.3.4,		
@RM: Job Scheduling	7.2.3.8,		
@RM: Resource Management	7.2.3.9),	@RM: Processing Effectiveness	6.4.23]

The **Suite III Multi-user Benchmark** for Unix systems measures the multi-user system performance of multiple systems under a variety of application mixes and resource loads, compares the multi-user performance of up to 20 single or multiprocessor systems at a time, predicts the effects of adding users to a system under a given application mix and resource load, and assesses the multi-user performance of a single system under varying use conditions. The Multi-user Benchmark uses 31 functional tests to evaluate system performance under user-defined application mixes, consisting of the following operation types: accounting, compiling, database management, graphics, scientific operations, spreadsheet, user interface operations, and word processing.

[(@RM: Text Editing	7.1.1.1,		
@RM: Graphics Editing	7.1.1.3,		
@RM: Pre & User-Defined Forms	7.1.2.3,		
@RM: Graphics Generation	7.1.5,		
@RM: Compilation	7.1.6.7,		
@RM: Database Management	7.2.1.1,		
@RM: Input/Output Support	7.2.3.2,		
@RM: Math/Statistics	7.2.3.4,		
@RM: Job Scheduling	7.2.3.8,		
@RM: Resource Management	7.2.3.9),	@RM: Processing Effectiveness	6.4.23]

The **Unix Utility Benchmark, Milestone**, uses actual Unix tools to perform different tasks repetitively. It is immune to excessive CPU instruction and data caching, and is functionally representative of real user loads. In simulating different user types, Milestone invokes standard Unix utilities that are typically performed by the specified user. The user types are: administrative assistant, calculator/spreadsheet, database user, manager,

scientist, software engineer, text processing.

[(@RM: Pre & User-Defined Forms	7.1.2.3,		
@RM: Sort/Merge	7.1.4,		
@RM: Linking/Loading	7.1.6.13,		
@RM: Database Management	7.2.1.1,		
@RM: File Management	7.2.1.3,		
@RM: Electronic Mail	7.2.1.4,		
@RM: Command Language Processing	7.2.3.1,		
@RM: Input/Output Support	7.2.3.2,		
@RM: Kernel	7.2.3.3,		
@RM: Math/Statistics	7.2.3.4,		
@RM: Resource Management	7.2.3.9,		
@RM: Comparison	7.3.1.1,		
@RM: Spelling Checking	7.3.1.2),	@RM: Processing Effectiveness	6.4.23]

The **X Windows Benchmark, Suite X,** provides over 140 specific tests that test performance in each of the X Operation Categories as well as the Primary X User Environments. The X Operation Categories are: graphics, including lines, polygons, circles, etc.; text, including fixed and variable fonts; pixmaps, including icon sized and larger; and windows, including creating, deleting, and circulating. The Primary X User Environments are: engineering with CAD/CAM and CAE; desktop publishing; and software development.

[(@RM: Text Editing	7.1.1.1,		
@RM: Graphics Editing	7.1.1.3,		
@RM: Pre & User-Defined Forms	7.1.2.3,		
@RM: Graphics Generation	7.1.5,		
@RM: Input/Output Support	7.2.3.2,		
@RM: Resource Management	7.2.3.9),	@RM: Processing Effectiveness	6.4.23]

The **Suite II Subsystem Benchmark** measures the performance of seven basic hardware and software subsystem areas: math, floating point, memory, function calls, system calls, disk, and inter-process communication. It does this while comparing and reporting on the performance of up to 20 systems.

[(@RM: Input/Output Support	7.2.3.2,		
@RM: Kernel	7.2.3.3,		
@RM: Math/Statistics	7.2.3.4,		
@RM: Job Scheduling	7.2.3.8,		
@RM: Resource Management	7.2.3.9),	@RM: Processing Effectiveness	6.4.23]

Finally, the **AIM Performance Reports (APR)** provide Unix users with a simple method of assessing the relative performance of Unix systems, from PCs to mainframes. AIM benchmarks (see Suites II and III and Milestone, above) are run on the target machines to test system performance. APR subscribers receive regular updates that contain technical information on the performance of the latest Unix systems. Subscribers can use the reports to make price/performance comparisons and to select systems for further evaluation.

Primary References: [Roybal 1990] K. Roybal, "AIM Procurement Guide," AIM Technology, Version 1.0, 1990.

Host/OS: Unix and OS/2

Vendors/Agents: [AIM]

Method: Automated test suites.

Inputs: The AIM test suites, operating system, utilities, and host computer.

Process:
1. Obtain the AIM test suite(s).
2. Run the tests.
3. Collect and analyze the results.

Outputs: Report(s) analyzing the strengths and weaknesses of the tested subsystem(s).

9.6 TRADE JOURNAL OPERATING SYSTEM SHELL EVALUATIONS

Purpose: Various trade journals have evaluated and compared operating system (OS) shells. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of OS shells, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: Text Editing	7.1.1.1,	(@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9);
@RM: File Management	7.2.1.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Command Language Processing	7.2.3.1,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
(@RM: Sort/Merge	7.1.4,		
@RM: Input/Output Support	7.2.3.2,		
@RM: Math/Statistics	7.2.3.4,		
@RM: Job Scheduling	7.2.3.8,		
@RM: Resource Management	7.2.3.9,		
@RM: Comparison	7.3.1.1),	@RM: Completeness	6.4.9]

Primary References: [Brown 1990] B. Brown, "Playing the DOS Shell Game," *PC Magazine*, vol. 9, no. 11, 12 June 1990, pp. 185-244.

[Marshall 1990] P. Marshall, Product Comparison, "Giving DOS a New Face," *InfoWorld*, vol. 12, issue 16, 16 April 1990, pp. 57-77.

Vendors/Agents: [PC Magazine, InfoWorld]

Method: Capabilities checklist(s).

Inputs: Checklists, OS shell(s), and documentation.

Process: Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed checklist(s).

9.7 TRADE JOURNAL OPERATING SYSTEM UTILITY EVALUATIONS

Purpose: Various trade journals have evaluated and compared operating system (OS) utilities. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of OS utilities, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: File Management	7.2.1.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33);
@RM: Input/Output Support	7.2.3.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35);
@RM: Import/Export	7.2.3.6,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35);
@RM: Emulation	7.3.2.13,	(@RM: Completeness	6.4.9,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: System Compatibility	6.4.35);
(@RM: Text Editing	7.1.1.1,		
@RM: Sort/Merge	7.1.4,		
@RM: Compression	7.1.6.16,		
@RM: Encryption	7.1.6.18,		
@RM: Math/Statistics	7.2.3.4,		
@RM: Job Scheduling	7.2.3.8),	@RM: Completeness	6.4.9]

Primary References: [Lauriston 1990b] R. Lauriston, "Hard Disk Health Insurance," *PC World*, vol. 8, no. 7, July 1990, pp. 109-118.

[Goodwin 1990] M. Goodwin, "Disk Trouble? No, Thanks," *PC World*, vol. 8, no. 7, July 1990, pp. 133-148.

[Walkenbach 1990a] J. Walkenbach, "No-Excuses Backup Software," *PC World*, vol. 8, no. 7, July 1990, pp. 149-164.

[Mendelson 1990] E. Mendelson, "Disaster Relief: DOS Utilities Save the Day," *PC Magazine*, vol. 9, no. 6, 27 March 1990, pp. 97-132.

[Lauriston 1990] R. Lauriston, "Cache Values," *PC World*, vol. 8, no. 2, February 1990, pp. 130-139.

[Mendelson 1989] E. Mendelson, "Backup Software: For the Moment After," *PC Magazine*, vol. 8, no. 8, August 1989, pp. 269-322.

Vendors/Agents: [PC World, PC Magazine]

Method: Capabilities checklist(s) and performance benchmark tests.

Inputs: Checklists, benchmark tests, OS utilities, and documentation.

Process: 1) Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.
2) Run the performance benchmark tests.

Outputs: Completed checklist(s) and performance reports.

9.8 TRADE JOURNAL EXPERT SYSTEM SHELL EVALUATIONS

Purpose: Various trade journals have evaluated and compared expert system shells. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of expert system shells, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: Program Generation	7.1.7.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Math/Statistics	7.2.3.4,	@RM: Completeness	6.4.9]

Primary References: [Brody 1989] A. Brody, Product Comparison, "The Experts," *InfoWorld*, vol. 11, issue 25, 19 June 1989, pp. 59-75.

Vendors/Agents: [InfoWorld]

Method: Capabilities checklist(s).

Inputs: Checklists, expert system shell(s), and documentation.

Process: Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed checklist(s) and performance reports.

10. REQUIREMENTS/DESIGN SUPPORT ASSESSORS

These assessors measure the suitability and effectiveness of various software definition, specification, and design tools. This specifically includes assessors of Ada Program Design Language (PDL) implementations and/or guidelines in the use of Ada as a PDL.

10.1 SEI DESIGN SUPPORT EXPERIMENT

Purpose: Evaluation of the design and code development capabilities of an environment, as represented in a small project. An experiment was designed to simulate the activities normally associated with small projects, namely the design, creation, modification, and testing of a single unit or module. See also the SEI Debugging or Unit Testing Experiments [7.9 or 8.2].

[@RM: Preliminary Design	7.1.6.4,	(@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Storage Effectiveness	6.4.32);
@RM: Detailed Design	7.1.6.5,	(@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Storage Effectiveness	6.4.32)]

Primary References: [Weiderman 1987] N.H. Weiderman and A.N. Habermann, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, Chapter 5, DTIC Number AD A180 905.

[Weiderman: Evaluation of Ada Environments 4.13]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment.

Inputs: The "generic" experiment description, an APSE, and host computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in checklist of functional elements present and missing, tables of time and space data, and subjective judgments based on the experience.

10.2 REQUIREMENTS PROTOTYPING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of requirements prototyping by developing a list of functional capabilities.

[@RM: Requirements Prototyping 7.3.2.2, @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 10.2-1) and requirements prototyping documentation.

Process: Check off capabilities demonstrated by the requirements prototyping system or discussed in the documentation.

Outputs: A list of capabilities provided by the requirements prototyping system.

Table 10.2-1 Requirements Prototyping Capabilities Checklist

FEATURE	FOUND	NOTES
Standards Requirements Libraries Executable Specifications Fourth Generation Languages or Very High Level Languages Reusable Building Blocks and Associated Tools Man-Machine Interface Prototyping Capabilities Applications Generators Previous Software Version Import Capabilities		

10.3 SIMULATION AND MODELING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of simulation and modeling by developing a list of functional capabilities.

[@RM: Simulation and Modeling 7.3.2.3, @RM: Completeness 6.4.9]

Primary References: [ISTAR 1987] Workshop on Future Development Environments, Information Science and Technology Assessment for Research, Conference on Information Mission Area (IMA) Productivity, Department of Army Director of Information Systems for Command, Control, Communications and Computers, 13-15 April 1987, pp 28.

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 10.3-1) and simulation and modeling documentation.

Process: Check off capabilities demonstrated by the simulation and modeling system or discussed in the documentation.

Outputs: A list of capabilities provided by the simulation and modeling system.

Table 10.3-1 Simulation and Modeling Capabilities Checklist

FEATURE	FOUND	NOTES
Conceptual Modeling Support		
Domain-Specific Knowledge Base		
Inferencing Systems		
Operational Environment Modeling Support		
User Modeling Support		
Model Browsers		
Game and Risk Models Database		
Functional Allocation Methodologies Database		
Scaling Rules Database		
Constraint Evaluation Tools		
Precision Estimators		
Computer System Modeling		
Interface/Input Support		
Graphical		
Menus		
Tabular		
Command		
Model Subject		
Control Flow		
Information Flow		
Human/Machine Procedures		
Outputs		
Response Time Estimates		
Throughput Estimates		
Resource Utilization Estimates		
System Types Supported		
Real Time		
Distributed		
Multiprocessor		
Standard Computer System Models Database		
Preprogrammed Models of Common Distributed System Functions		
Underlying Mathematical Theory (e.g., Queueing Network Theory)		

10.4 NADC/SPS CASE TOOLS EVALUATION

Purpose: Development of evaluation criteria and evaluation of selected candidate methods and tools. The focus of this investigation was Computer Aided Software Engineering (CASE) tools and methods applied during the early life cycle phases/activities (system and software requirements and top-level design) and applied to large, time-critical systems. Three commercially available CASE tools were selected for evaluation, following an initial survey of more than 100 possibilities. An experiment based on a sample problem (submarine detection with a sonobouy) was created and carried out using three systems. Evaluation criteria, detailed questions, and a scoring system were developed and applied in three areas: method, automation, and vendor support.

[@RM: System Requirements	7.1.6.1,	(@RM: Augmentability	6.4.4,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: System Clarity	6.4.34,
		@RM: Training	6.4.37);
@RM: Software Requirements	7.1.6.2,	(@RM: Augmentability	6.4.4,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: System Clarity	6.4.34,
		@RM: Training	6.4.37);
@RM: Preliminary Design	7.1.6.4,	(@RM: Augmentability	6.4.4,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: System Clarity	6.4.34,
		@RM: Training	6.4.37)]

Primary References: [Donaldson 1988] C. Donaldson and P.B. Dyson, "Computer-Aided Systems and Software Engineering Products for Time-Critical Applications Development," Software Productivity Solutions (SPS), Inc., April 1988.

[Stuebing 1988] H.G. Stuebing, "Evaluation of Computer Aided Systems/Software Engineering Products for Time-Critical Naval Systems," *Proceedings of the Conference Methodologies and Tools for Real Time Systems*, November 14-15, 1988.

Host/OS: Various Workstations (PC/AT, Apollo, Sun, VAXStation)

Vendors/Agents: [NADC, SPS]

Method: Structured Experiment

Inputs: Evaluation criteria and questions, sample problem definition, and candidate methods and tools

Process: For each candidate method/tool, carry out development of software requirement specification and top-level design documents. Answer evaluation questions and fill out scoring sheets.

Outputs: Evaluation reports containing two levels of detail: executive summaries with top-level scoring and detailed descriptions and analyses with questions, answers, and individual scores.

10.5 TIME-CRITICAL APPLICATIONS SUPPORT CHECKLIST

Purpose: Evaluation of the completeness of time-critical applications support by developing a list of functional capabilities.

[(@RM: System Requirements 7.1.6.1,
@RM: Software Requirements 7.1.6.2,
@RM: Preliminary Design 7.1.6.4,
@RM: Detailed Design 7.1.6.5), @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Checklist.

Inputs: Capabilities checklist (see Table 10.5-1) and time-critical applications support documentation.

Process: Check off capabilities demonstrated by the time-critical applications support or discussed in the documentation.

Outputs: A list of capabilities provided by the time-critical applications support.

Table 10.5-1 Time-critical Applications Support Checklist

FEATURE	FOUND	NOTES
Periodic and aperiodic events		
Synchronization of sequential and concurrent processes		
Execution sequence of components		
Timing constraints for events, sequences of events, processes, and sequences of processes		
Precision of a system's response to internal and external events		
Interrupts and the extent of process context switching		
Processing of discrete and time continuous data		
Allocation of critical timelines and resource utilizations		
Data throughput		
Task priority changes due to system mode changes or failures		
Task management (time-slicing, run-to-completion)		
Graphical time-line depiction for tasks, showing dependencies and concurrencies		
Simulations for the host		
Dynamic analysis for the target		

10.6 STEM/DRAPER REQUIREMENTS/DESIGN TOOLS EVALUATION

Purpose: Evaluation of the capabilities of CASE tools that support requirements analysis and/or design. The primary goal of this project is to develop test and evaluation procedures and apply them to CASE tools. Secondary goals are to compile a list of CASE tools, to speed up technology insertion in the use of CASE tools, and to contribute to the STEM database [@STSC 1990]. The evaluation procedures developed under this project have emphasized creation of a hierarchical checklist. The top-level items in the checklist include the following "selection criteria areas:"

Information Capture
Methodology Support
Model Analysis
Requirements Tracing
Programming Language Support
Data Repository
Documentation
Data Import/Export
Reusability Support
User Interface
Vendor Information
Multi User Environment
Configuration Control
Host Platform
Costs
Other

The cited reference provides second-level checklist items under each of the above selection criteria areas.

[@RM: Software Requirements	7.1.6.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Cost	6.4.11,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Granularity	6.4.17,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: Simplicity	6.4.30,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Preliminary Design	7.1.6.4,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Cost	6.4.11,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Granularity	6.4.17,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,

		@RM: Required Configuration	6.4.27,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: Simplicity	6.4.30,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Detailed Design	7.1.6.5,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Cost	6.4.11,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Granularity	6.4.17,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: Self-Descriptiveness	6.4.29,
		@RM: Simplicity	6.4.30,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38)]

Primary References: [Van Buren 1990] J.K. Van Buren, "Evaluation Procedures for Requirements Analysis and Design CASE Tools," USAF STSC HQ USAF/SC Joint Software Conference Proceedings, Salt Lake City, 23-26 April 1990 [@STSC 1990].

[Software Tool Evaluation Model (STEM) 4.20]

Vendors/Agents: [STSC, Draper]

Method: Capabilities Checklist

Inputs: Capabilities checklist and CASE tool and its documentation.

Process: Checklist off capabilities demonstrated by testing or discussed in the documentation.

Outputs: A list of capabilities provided by the tool.

10.7 SOFTWARE METHODOLOGY CATALOG

Purpose: "This catalog provides a consolidated reference for methods used over the total spectrum of software development. A primary objective is to provide a brief overview of each method, and to give for each method some insight into its underlying assumptions, the software development activities which it supports, and other characteristics associated with its use. ... One final objective is to provide a facility for contrasting the various methods relative to selected attributes."

"... The catalog represents an effort to obtain the most current information available for each method. To this end, surveys were developed, and responses were solicited from developers. In writing the catalog the authors have elected to serve as reporters rather than evaluators. Evaluative statements that appear in the catalog are based upon survey responses."

"The use of the term method in this catalog needs further explanation. Both tools and approaches exist over wide spectra. Within these spectra, making a clear distinction between a sophisticated tool and a method on the one hand, or between a method and a prescriptive approach on the other, is difficult. Thus, the authors' use of the term **method** in this catalog has been widened to include schema that could be termed tools or approaches."

Although the authors state that the catalog covers "the total spectrum of software development," most of the methods described within deal, primarily, with the front end activities of software development from requirements engineering through detailed design. Each method is presented in a common format containing: background, description, technical aspects, project control and communication, ease of use, acquisition factors, and references. The criteria that are used as a basis for method comparison are: coverage and prescriptiveness, robustness, expressiveness, analyzability and stability, correctness and effectiveness, manageability, productivity, and ease of adoption.

[@RM: Software Requirements	7.1.6.2,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Application Independence	6.4.3,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Consistency	6.4.10,
		@RM: Cost	6.4.11,
		@RM: Maturity	6.4.19,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Required Configuration	6.4.27,
		@RM: Retargetability	6.4.28,
		@RM: Software Production Vehicle(s)	6.4.31,
		@RM: Traceability	6.4.36,
		@RM: Training	6.4.37,
		@RM: Vendor Support	6.4.38,
		@RM: Visibility	6.4.40);
@RM: Preliminary Design	7.1.6.4,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,

		@RM: Application Independence 6.4.3,
		@RM: Capacity 6.4.6,
		@RM: Commonality 6.4.7,
		@RM: Communication Effectiveness 6.4.8,
		@RM: Completeness 6.4.9,
		@RM: Consistency 6.4.10,
		@RM: Cost 6.4.11,
		@RM: Maturity 6.4.19,
		@RM: Operability 6.4.21,
		@RM: Power 6.4.22,
		@RM: Processing Effectiveness 6.4.23,
		@RM: Proprietary Rights 6.4.24,
		@RM: Required Configuration 6.4.27,
		@RM: Retargetability 6.4.28,
		@RM: Software Production Vehicle(s) 6.4.31,
		@RM: Traceability 6.4.36,
		@RM: Training 6.4.37,
		@RM: Vendor Support 6.4.38,
		@RM: Visibility 6.4.40);
@RM: Detailed Design	7.1.6.5,	(@RM: Accuracy 6.4.1,
		@RM: Anomaly Management 6.4.2,
		@RM: Application Independence 6.4.3,
		@RM: Capacity 6.4.6,
		@RM: Commonality 6.4.7,
		@RM: Communication Effectiveness 6.4.8,
		@RM: Completeness 6.4.9,
		@RM: Consistency 6.4.10,
		@RM: Cost 6.4.11,
		@RM: Maturity 6.4.19,
		@RM: Operability 6.4.21,
		@RM: Power 6.4.22,
		@RM: Processing Effectiveness 6.4.23,
		@RM: Proprietary Rights 6.4.24,
		@RM: Required Configuration 6.4.27,
		@RM: Retargetability 6.4.28,
		@RM: Software Production Vehicle(s) 6.4.31,
		@RM: Traceability 6.4.36,
		@RM: Training 6.4.37,
		@RM: Vendor Support 6.4.38,
		@RM: Visibility 6.4.40)]

Primary References: [Von Gerichten 1989] L. Von Gerichten, et al., "Software Methodology Catalog," CECOM Center for Software Engineering, US Army Communications-Electronics Command, C01-091JB-0001-01, March 1989, DTIC Number AD A210 548.

Vendors/Agents: [DTIC, CECOM]

Method: Questionnaire.

Inputs: Questionnaire (from catalog) and method documentation.

Process: Answer questions based on capabilities demonstrated by the method (tool), reading the documentation, or asking the vendor.

Outputs: Completed questionnaire.

10.8 CECOM: PROCEDURES FOR COMPUTER-AIDED SOFTWARE ENGINEERING TOOL ASSESSMENT

Purpose: "The purpose of this study is to develop procedures for the assessment of computer-aided software engineering (CASE) tools for tactical embedded systems. ... Due to the large number of CASE tools that are available currently, it is imperative for an organization about to purchase a tool to gain an understanding of the currently available tools and assess their attributes. Understanding what a tool does and comparing it to similar tools are difficult tasks, some of the reasons being:

- the diversity of functionality that exists among the tools,
- vendor claims for specific tools and the performance of these tools in actual practice are often dramatically different.

Hence, it is very important to develop criteria and procedures for the assessment of CASE tools that address parts as well as the whole software development life cycle."

Although there is some discussion of tool support for testing and maintenance, the reports focus, primarily, on the front end of the software life cycle from requirements engineering through code generation. The first report gives the evaluation criteria and assessment procedures to be used for CASE tools. The second report uses the criteria and procedures to assess eleven tools currently available.

[@RM: Anomaly Management			6.4.2;
@RM: Documentation Quality			6.4.13;
@RM: Functional Scope			6.4.15;
@RM: Operability			6.4.21;
@RM: Processing Effectiveness			6.4.23;
@RM: Proprietary Rights			6.4.24;
@RM: Software Production Vehicle(s)			6.4.31;
@RM: System Compatibility			6.4.35;
@RM: Training			6.4.37;
@RM: Vendor Support			6.4.38;
(@RM: Text Editing	7.1.1.1,		
@RM: Graphics Editing	7.1.1.3),	@RM: Power	6.4.22;
@RM: MIL-STD Format	7.1.2.1,	(@RM: Commonality	6.4.7,
		@RM: System Compatibility	6.4.35);
@RM: Software Requirements	7.1.6.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: Virtuality	6.4.39,
		@RM: Visibility	6.4.40);
@RM: Preliminary Design	7.1.6.4,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: Virtuality	6.4.39,
		@RM: Visibility	6.4.40);

@RM: Detailed Design	7.1.6.5,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: Virtuality	6.4.39,
		@RM: Visibility	6.4.40);
@RM: Program Generation	7.1.7.3,	(@RM: Completeness	6.4.9,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Traceability	6.4.36,
		@RM: Visibility	6.4.40);
@RM: Database Management	7.2.1.1,	(@RM: Completeness	6.4.9,
		@RM: Distributedness	6.4.12,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35);
(@RM: Requirements Reconstruction	7.1.7.2,		
@RM: Decompilation	7.1.7.5,		
@RM: Disassembling	7.1.7.6,		
@RM: Source Code Restructuring	7.1.7.8,		
@RM: Design Reconstruction	7.1.7.9,		
@RM: Import/Export	7.2.3.6,		
@RM: Access Control	7.2.3.7,		
@RM: Traceability Analysis	7.3.1.6,		
@RM: Completeness Checking	7.3.1.12,		
@RM: Consistency Checking	7.3.1.13,		
@RM: Reachability Analysis	7.3.1.16,		
@RM: Structure Checking	7.3.1.26,		
@RM: Random Test Generation	7.3.1.32,		
@RM: Requirements Prototyping	7.3.2.2,		
@RM: Simulation and Modeling	7.3.2.3,		
@RM: Design Prototyping	7.3.2.4,		
@RM: Coverage/Frequency Analysis	7.3.2.8,		
@RM: Real-Time Analysis	7.3.2.17),	@RM: Completeness	6.4.9;
(@RM: Cost Estimation	7.2.2.1,		
@RM: Scheduling	7.2.2.3,		
@RM: Work Breakdown Structure	7.2.2.4,		
@RM: Resource Estimation	7.2.2.5,		
@RM: Tracking	7.2.2.6,		
@RM: Configuration Management	7.2.2.7),	@RM: System Accessibility	6.4.33]

Primary References: [CECOM 1989] "Procedures For Computer-Aided Software Engineering Tool Assessment," CECOM Center for Software Engineering, US Army Communications-Electronics Command, ?, April 1989, DTIC Number pending.

[CECOM 1989a] "Evaluation Of Existing CASE Tools For Tactical Embedded Systems," CECOM Center for Software Engineering, US Army Communications-Electronics Command, ?, April 1989, DTIC Number pending.

Vendors/Agents: [CECOM]

Method: Questionnaire.

Inputs: Questionnaire (from first report), tool, and tool documentation.

Process: Answer questions based on capabilities demonstrated by the tool, reading the documentation, or asking the vendor.

Outputs: Completed questionnaire.

10.9 IEEE: AN EVALUATION OF CASE TOOLS

Purpose: "This paper presents the results of a study to select a workstation based computer aided software engineering (CASE) tool for a large software development concern. ... The paper presents the functions typically performed by CASE tools ... The paper then describes the approach used to evaluate and compare these tools. This approach involved development of a detailed product requirement list, a ranking of the importance of the requirements by potential users of the tool, and a relative rating (of) the most comprehensive of the available products according to the degree to which they satisfy the requirements. The user requirement rankings and the product ratings are then used to select the best available product."

[@RM: Software Requirements	7.1.6.2,	(@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Cost	6.4.11,
		@RM: Operability	6.4.21,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38);
@RM: Preliminary Design	7.1.6.4,	(@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Cost	6.4.11,
		@RM: Operability	6.4.21,
		@RM Required Configuration	6.4.27,
		@RM Vendor Support	6.4.38);
@RM: Detailed Design	7.1.6.5,	(@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Cost	6.4.11,
		@RM: Operability	6.4.21,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38)]

Primary References: [Troy 1987] D.A. Troy, "An Evaluation of CASE Tools," Proceedings of the 1987 International Computer Software and Applications Conference, IEEE Computer Society, pp. 124-130, 1987.

Vendors/Agents: [IEEE Computer]

Method: Checklist.

Inputs: Checklist, tool, and tool documentation.

Process: Rate features demonstrated by the tool, or described by tool documentation or discussions with vendor.

Outputs: Completed checklist.

10.10 ACM SIGSOFT: SELECTION CRITERIA FOR ANALYSIS AND DESIGN CASE TOOLS

Purpose: "Computer Aided Software Engineering (CASE) has been receiving increasing attention because of its potential for substantial productivity improvement of software development. Selecting CASE tools best suited to organizations needs can be a challenge because of wide variations among CASE features, completeness, terminology, and usage characteristics. This paper presents specific performance criteria of CASE tools and ranks them as required versus "nice to have." It is based on an in depth investigation of several packages."

[@RM: Software Requirements	7.1.6.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Traceability	6.4.36);
@RM: Preliminary Design	7.1.6.4,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Traceability	6.4.36);
@RM: Detailed Design	7.1.6.5,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Traceability	6.4.36)]

Primary References: [Baram 1989] G. Baram and G. Steinberg, "Selection Criteria for Analysis and Design CASE Tools," Software Engineering Notes, ACM SIGSoft, Vol. 14, No. 6, pp. 73-80, October 1989.

Vendors/Agents: [ACM]

Method: Checklist.

Inputs: Checklist, tool, and tool documentation.

Process: Rate features demonstrated by the tool, or described by tool documentation or discussions with vendor.

Outputs: Completed checklist.

10.11 TRADE JOURNAL REQUIREMENTS AND DESIGN TOOL EVALUATIONS

Purpose: Various trade journals have evaluated and compared requirements and design tools. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of requirements and design tools, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: Software Requirements	7.1.6.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Distributedness	6.4.12,
		@RM: Functional Overlap	6.4.14,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Training	6.4.37,
		@RM: Vendor Support	6.4.38);
@RM: Preliminary Design	7.1.6.4,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Distributedness	6.4.12,
		@RM: Functional Overlap	6.4.14,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Training	6.4.37,
		@RM: Vendor Support	6.4.38);
@RM: Detailed Design	7.1.6.5,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Distributedness	6.4.12,
		@RM: Functional Overlap	6.4.14,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Training	6.4.37,
		@RM: Vendor Support	6.4.38);
@RM: Design Generation	7.1.7.1,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Program Generation	7.1.7.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);

(@RM: Traceability Analysis	7.3.1.6,		
@RM: Completeness Checking	7.3.1.12,		
@RM: Consistency Checking	7.3.1.13,		
@RM: Syntax & Semantics Check	7.3.1.15,		
@RM: Requirements Simulation	7.3.2.1,		
@RM: Requirements Prototyping	7.3.2.2,		
@RM: Simulation and Modeling	7.3.2.3,		
@RM: Design Prototyping	7.3.2.4),	@RM: Completeness	6.4.9]

Primary References: [Sullivan-Trainor 1990] M.L. Sullivan-Trainor, Product Spotlight, "Buyer's Scorecard," *ComputerWorld*, vol. 24, no. 15, 9 April 1990, pp. 68-69.

[Fersko-Weiss 1990] H. Fersko-Weiss, "CASE Tools for Designing Your Applications," *PC Magazine*, vol. 9, no. 2, 30 January 1990, pp. 213-251.

[McClure 1989] C. McClure, "The CASE Experience," *BYTE*, vol. 14, no. 4, April 1989, pp. 235-245.

Vendors/Agents: [ComputerWorld, PC Magazine, BYTE]

Method: Capabilities checklist(s).

Inputs: Checklists, requirements and design (CASE) tool(s), and documentation.

Process: Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed checklist(s).

11. CONFIGURATION MANAGEMENT SUPPORT ASSESSORS

These assessors examine the performance, usability, and completeness of the APSE or APSE component functionality related to controlling the contents of software systems. This includes monitoring the status, preserving the integrity of released and developing versions, and controlling the effects of changes throughout the lifetime of the software system.

11.1 CONFIGURATION MANAGEMENT CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of configuration management by developing a list of functional capabilities.

[@RM: Configuration Management 7.2.2.7, @RM: Completeness 6.4.9]

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist.

Inputs: Capabilities checklist (see Table 11.1-1) and configuration management documentation.

Process: Check off capabilities demonstrated by the configuration manager or discussed in the documentation.

Outputs: A list of capabilities provided by the configuration manager.

Table 11.1-1 Configuration Management Capabilities Checklist

FEATURE	FOUND	NOTES
Version Management Archive Protect Integrity Checking Support for Review Process Backup and Recovery Support Support for (Backward) Delta Storage Method Compression Encryption Support for Export/Import of Version Control Data Programmatic Interface		
Revision Management Support for Multiple Development Paths (Branching/Variations) Support for Reconciliation/Merging		
Text and Binary Support		
Usage Administration/Access Control/Security		
Audit Support		
Configuration Library Create Delete Verify		
Library Elements Create Delete Fetch/Check Out Read-Only Check Out Reserve/Lock Unreserve/Unlock Replace/Check In Differences		
Element Classes Create Delete Insert Element Remove Element		

Table 11.1-1 Configuration Management Capabilities Checklist (Continued)

FEATURE	FOUND	NOTES
Reports/Listings Elements Reservations/Check Outs History Annotation Completeness		
Level Control		
Test Control Procedures Data Results Failure Reporting		
Integration with Development Environment		
Activity Tracking		
User-Configurable Automated Software Builds		
Component Dependency Tracking		
Support for Expandable Keywords		
Operating System File Name Independence		
Networking Capabilities		
Search Capabilities		

11.2 SEI CONFIGURATION MANAGEMENT EXPERIMENT

Purpose: Evaluation of the configuration management and version control capabilities of an environment. An experiment was designed to simulate the system integration and testing phase of the life cycle by having three separate development lines of descent from a single baseline.

[@RM: Configuration Management	7.2.2.7,	(@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23)]

Primary References: [Weiderman 1987] N.H. Weiderman and A.N. Habermann, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, Chapter 3, DTIC Number AD A180 905.

[Weiderman: Evaluation of Ada Environments 4.13]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment.

Inputs: The "generic" experiment description, an APSE, and host computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in checklist showing functional elements present and missing, a table of elapsed-time values for certain specific operations, and subjective judgments based on the experience.

11.3 CONFIGURATION MANAGEMENT ASSESSMENT QUESTIONNAIRE

Purpose: The document presents a hierarchical breakdown of configuration management shown in Fig. 11.3-1. Requirements for each element in the hierarchy are listed for certain attributes. Each requirement is augmented by one or more questions which address the requirement.

[@RM: Configuration Management	7.2.2.7,	(@RM: Augmentability	6.4.4,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Consistency	6.4.10,
		@RM: Cost	6.4.11,
		@RM: Documentation Quality	6.4.13,
		@RM: Generality	6.4.16,
		@RM: Granularity	6.4.17,
		@RM: Maturity	6.4.19,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Rehostability	6.4.26,
		@RM: Storage Effectiveness	6.4.32,
		@RM: System Compatibility	6.4.35,
		@RM: Visibility	6.4.40)]

Primary References: [E&V Report 1984] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 1.0," 17 October 1984, Appendix B of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, November 1984, pp. B-86 - B-91, DTIC Number AD A153 609.

[Requirements for E&V 4.5]

Vendors/Agents: [E&V Team]

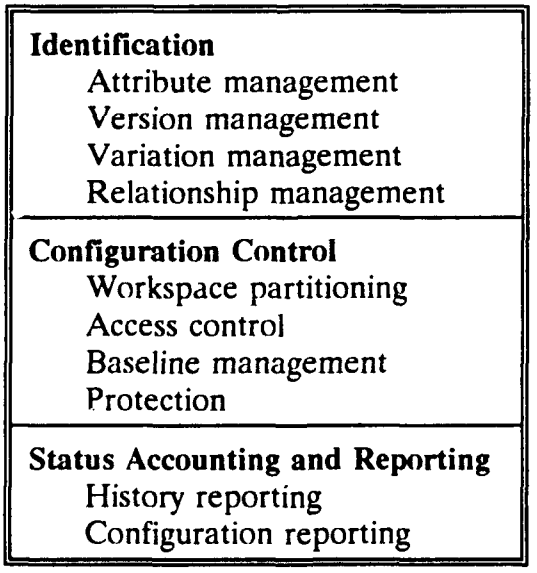

Figure 11.3-1 Configuration Management Hierarchy

Method: Questionnaire.

Inputs: Questionnaire and configuration management documentation.

Process: Answer questions based on documentation, using the configuration manager, or asking the vendor.

Outputs: Completed questionnaire.

11.4 Ada-EUROPE: PRODUCT MANAGEMENT QUESTIONNAIRE

Purpose: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@Buxton 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided. Chapter 16 discusses issues associated with project and product management (integration with a database, estimation planning and monitoring tools, quality management and configuration management, and social and legal aspects).

[@RM: Configuration Management 7.2.2.7, @RM: Completeness 6.4.9]

Primary References: [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986, Chapter 16.

[Ada-Europe: Selecting an Ada Environment 4.9]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire.

Inputs: Questionnaire, tool, and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed questionnaire.

12. DISTRIBUTED SYSTEMS DEVELOPMENT AND RUNTIME SUPPORT ASSESSORS

These assessors examine the ability of the APSE or APSE components to support software development for distributed processing systems, and to provide runtime support for distributed processing systems.

12.1 PERFORMANCE OF PARALLEL Ada

Purpose: Evaluation of run-time performance of multiprocessor systems. The two references cited below appear in the PIWG special issue of Ada Letters [@PIWG 1990]. The first reference, by Clapp and Mudge, is a general discussion of parallel and distributed systems and the problem of developing benchmark tests to measure their performance. The authors define parallel systems as "those in which program units are assigned processors at runtime" and distributed systems as "those in which program units are assigned processors at or before compile time."

The second reference, by Goforth, Collard, and Marquardt of NASA Ames Research Center, describes an example of the development and execution of benchmark tests applied to a specific parallel Ada system. The chosen Ada application was based on the NASA/NBS Standard Reference Model for Telerobot Control System Architecture [@Albus 1986]. Experiments were run on a Sequent Balance 8000 shared-memory multiprocessor computer. The number of processors was varied from one to 16. The reference includes plots of experimental results and a discussion of benchmarking difficulties unique to parallel systems.

[(@RM: Compilation 7.1.6.7, @RM: Runtime Environment 7.2.3.5), @RM: Processing Effectiveness 6.4.23]

Primary References: [Clapp 1990] R.M. Clapp and T. Mudge, "Parallel and Distributed Issues," pp. 33-37 of Ada Letters, Special Edition from SIGAda, the ACM Special Interest Group on Ada Performance Issues Working Group, Vol. X, Number 3, Winter 1990. [@PIWG 1990]

[Goforth 1990] A. Goforth, P. Collard, and M. Marquardt, "Performance Measurement of Parallel Ada: An Application Based Approach," pp. 38-58 of Ada Letters, Special Edition from SIGAda, the ACM Special Interest Group on Ada Performance Issues Working Group, Vol. X, Number 3, Winter 1990. [@PIWG 1990]

[Ada Performance Issues (PIWG Special Edition) 4.19]

Vendors/Agents: [NASA/Ames]

Method: Test suite.

Inputs: Source code, Ada compiler and runtime system, and host (and target) computer.
Process:

1. Obtain (or build similar) benchmark tests.
2. Compile and run tests.

Outputs: Reports on the outcome of each test run.

13. DISTRIBUTED APSE ASSESSORS

These assessors examine the ability of two or more distributed APSEs to communicate in cooperative ways in supporting the development of mission critical software at diverse geographical locations.

13.1 DISTRIBUTED APSE QUESTIONNAIRE

Purpose: Evaluation of the APSE's ability to be used in a distributed environment.

[@RM: Whole APSE Issues	3,	(@RM: Anomaly Management	6.4.2,
		@RM: Commonality	6.4.7,
		@RM: Communication Effectiveness	6.4.8,
		@RM: Consistency	6.4.10,
		@RM: Distributedness	6.4.12,
		@RM: Functional Overlap	6.4.14,
		@RM: Operability	6.4.21,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35);
@RM: Performance Monitoring	7.2.1.10,	@RM: Completeness	6.4.9;
@RM: Tracking	7.2.2.6,	@RM: Distributedness	6.4.12]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 13.1-1) and APSE documentation.

Process: Answer questions based on reading the documentation, using the APSE, or asking the vendor.

Outputs: Completed questionnaire.

Architecture

Type of Distribution

What is distributed on the APSE: processing resources, data, or both?

Heterogenous/Homogenous

Does the APSE support a heterogenous hardware configuration or is it restricted to implementation on a homogenous hardware configuration?

Is there special hardware required for its implementation on a heterogenous configuration?

Are there special software communication protocols that are required for implementation on a heterogenous configuration?

Node Transparency

Is the same toolset available on all nodes in the APSE?

If so, how is the commonality defined (e.g., common user interface, common functionality, and support by a common vendor)?

If not:

Is the user-interface and functionality the same across all nodes?

If so, is the definition tailorable?

If not, is this due to the APSE using its distributed nature to partition one type of user from another (e.g., designers on workstations and coders on minis)?

Does the vendor clearly specify the APSE functionality of each node and the expected target user?

Is the tool variance due to the particular operating system which a node supports?

Are all APSE data available to all nodes and tools?

Can the user access and use the APSE from any node without retraining or using a different set of commands?

Server Node

If there is a server node, how is it configured differently to handle the additional functions it performs?

Fault Tolerance, Isolation, and Recovery (Anomaly Management)

If a node goes down, are the functions and/or data at that node unavailable to the rest of the network until it comes back on line or are the functions and/or data duplicated on another node?

If duplicated, how current are they?

Performance Monitor

If a performance monitor is provided, can it balance the load on the nodes to maintain a certain performance level? If so:

Is the balancing automatically performed or is human interaction required?

If automatic, can it be delayed or prevented by human interaction? If so, how?

What method does it use for balancing and when is the balancing done?

Does the method support load-balancing across both homogenous and heterogenous implementations?

What are its limitations?

Is the balancing function non-intrusive as far as the user is concerned?

If not, how is the user affected during its execution?

Security

For each of the four levels (individual tool level, node level, whole APSE level, and underlying operating system level), do the discretionary access control features of the distributed APSE permit it to be certified as a Trusted Computer System Commercial Product?

At what division (A, B, or C) and at what class (e.g., C1 or C2) within the division as defined in the DoD Trusted Computer System Evaluation Criteria (also known as the Orange Book [@DoD 1983a]) is it certified?

Figure 13.1-1 Distributed APSE Questionnaire

Interprocessor Distribution/Communication

Are facilities provided to allow a process to be distributed? If so:
- Does the scheme support both homogenous and heterogenous interprocessor distribution?
- What is the performance cost?
  - If the cost is high, can the function be prevented or modified?
    - What are the consequences of preventing or modifying it?
- Which APSE components use interprocessor distribution?
  - Is the function required by the components?
    - If so, does the function require the availability of specific nodes or will any pair of nodes suffice?
    - If not, what are the intended benefits of interprocessor distribution?
- Does interprocessor distribution understand, interface with, and support computer security?
  - If so, how is the security implemented?
  - If not:
    - What aspects of the function would be concerned with computer security?
    - How does it affect the usability of the APSE?
    - What obstructions to security does the vendor perceive?
    - Are there plans for the function to include computer security in the future?
- What Operating System functions are required to support interprocessor distribution?
  - Can these functions be supplied by other software?
    - If so, is this achievable by the user or does it require the vendor?
      - If the vendor has to do it, is there an extra charge or is it covered under the annual maintenance fees:
- What deadlock prevention, race detection, etc. schemes are supported by interprocessor distribution?
- What is the scheduling algorithm used by interprocessor distribution?
- What communication software and hardware is required for interprocessor distribution?
- What protocols are used by this function?
  - Do the protocols support standard definitions (e.g., TCP/IP, Ethernet, and Mil-Std-1553)?
  - Can the protocols be user-defined?
  - Can one standard protocol be switched with another?
    - If so, can the switch be performed by the user or only by the vendor?
      - If it can only be done by the vendor, is there an extra charge for this or is it covered under the annual maintenance fees?
  - If more than one protocol is used, is there a clear, structure level definition for the protocols (e.g., TCP/IP for all system level communiques and Ethernet for all node level communiques)?
  - Do the user's manuals explicitly state how, where, and when these protocols are used?
  - Does the APSE provide all the software and hardware necessary to support the required protocols?

Project Management

Do the project management functions reside in one central location, performing data collection across the network? If so:
- Is there a subset of the project management toolset which resides on each node to perform the collection and transferral of information from that node to the central location?
- What software and/or hardware is required to support the communication of the distributed functionality?
- What is the protocol (or method) which is used by the toolset to communicate with separate functions or locations within itself?
- Is the project management data collection performed across all nodes automatically or does it require human interaction?

Does the project management function require all nodes in the APSE to be active in order for it to work?
- If not, what happens when a node goes down?

Can users dump or load information to or from the project management toolset from any node?
- If so, can it be done in batch mode or does it require interactive assistance from the user?

Can other tools (e.g., spreadsheets) access/load information to or from the project management toolset?
- If so, can this be done in batch mode or does it require interactive assistance from the user?

Figure 13.1-1 Distributed APSE Questionnaire (Continued)

14. "WHOLE APSE" ASSESSORS

These assessors examine or measure the overall quality or performance of an APSE considered as a whole rather than as a collection of individual parts individually assessed. A specific whole-APSE assessor may be designed to achieve a limited objective. An example of a limited objective is: evaluate the quality of an APSE in supporting a team of software developers performing a specific life cycle phase or activity such as preliminary design or integration testing. The results of such an evaluation could then become one ingredient of an integrated whole-APSE assessment (as described in Section 3.3), which has a broad objective.

14.1 APSE CHARACTERIZATION

Purpose: The purpose of this form is to provide an overview or summary of the capabilities and features of an APSE. This form can be used as an initial information gathering device to begin the process of whole-APSE assessment. This information would then be supplemented by results of detailed evaluations or examinations of attributes that are of specific interest to the potential buyer or user of an APSE.

[@RM: Whole APSE Issues	3,	(@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Cost	6.4.11,
		@RM: Maturity	6.4.19,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38)]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Blank APSE characterization form (see Fig. 14.1-1) and APSE documentation.

Process:
1. Complete the APSE characterization form
2. Select APSEs for further investigation based on information gathered from step 1.

Outputs: Completed APSE characterization form.

Name/Acronym:
Vendor: Address: Phone Number:
Cost ($, no charge, not available/applicable): Purchase _____ Seminars _____ Maintenance _____ In-house Classes _____ Documentation _____ Educational Videos _____ On-Line Help _____ On-Line Tutorials _____ Hot-Line Support _____
Problem Reporting/Resolution Procedures:
Frequency of Updates:
Usage Limitations (License Restrictions):
Host/Target(s) - Required Configurations:
Peripherals Supported:
Languages Supported & Interoperability Features:
Summary of Features:

Figure 14.1-1 APSE Characterization Form

Life Cycle Support - Capabilities/Major Activity:
Methodology Support:
Management Support:
Application-Specific Capabilities:
Documentation Support (editors, word processors, document generators, desktop publishing):
File/Database/Program Library Management (hierarchical, relational):
Access Control—Level of Granularity:
Integration Mechanism (standard file structures, database, standard intertool interfaces):
User Interface (command language, menus, icons)—Flexibility vs. Consistency:
Extensibility:
Support for Distributed Development:
Capacity (number of users, size of project):

Figure 14.1-1 APSE Characterization Form (Continued)

Typical Usage Scenarios (expertise of users, roles):
Developer:
Production Process/Vehicles:
Date First Released:
Previous Use:
References (documentation, evaluation results, case histories):

Figure 14.1-1 APSE Characterization Form (Continued)

14.2 Ada-EUROPE Ada ENVIRONMENT QUESTIONNAIRES

Purpose: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@Buxton 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided.

[@RM: Whole APSE Issues	3,	(@RM: Augmentability	6.4.4,
		@RM: Capacity	6.4.6,
		@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Proprietary Rights	6.4.24,
		@RM: Required Configuration	6.4.27,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Training	6.4.37)]

Primary References: [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986.

[Ada-Europe: Selecting an Ada Environment 4.9]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaires.

Inputs: Questionnaires (see Table 14.2-1), APSE, and APSE documentation.

Process: Answer the questions by using the APSE, reading the documentation, or asking the vendor of the APSE.

Outputs: Completed questionnaires.

Table 14.2-1 Ada-Europe Environment Questionnaires

ATTRIBUTE	RELEVANT SECTION(S) FROM LYONS' BOOK
Augmentability	7.2, 7.4, 8., 9.
Capacity	18.1
Commonality	8.
Completeness	4., 5., 7.1, 7.3, 9., 13.-17.
Operability	10.-12., 18.10-18.12
Power	10.3-10.6
Processing Effectiveness	18.2-18.19
Proprietary Rights	19.
Required Configuration	2., 3., 18.2, 18.3
System Accessibility	6.
System Compatibility	8.
Training	10.2

14.3 CROSS-DEVELOPMENT SYSTEM SUPPORT QUESTIONNAIRE

Purpose: Evaluation of the APSE's ability to support the development of an application on a host computer for implementation on a different target computer, where the target computer is usually incapable of compiling, linking, and debugging software.

[(@RM: Whole APSE Issues	3,		
@RM: Assembling	7.1.6.6,		
@RM: Compilation	7.1.6.7,		
@RM: Linking/Loading	7.1.6.13,		
@RM: Simulation and Modeling	7.3.2.3,		
@RM: Debugging	7.3.2.5,		
@RM: Emulation	7.3.2.13,		
@RM: Timing Analysis	7.3.2.14),	@RM: Completeness	6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 14.3-1) and APSE documentation.

Process: Answer questions based on documentation, using the tools, or asking the vendor.

Outputs: Completed questionnaire.

Transformation

Are there target-optimizing cross-assemblers?
Does the front end support multiple code generators?
What language features are supported by the code generator?

Where are the pragmas defined?
- Are they all defined and understood by the front end?
- Are they all defined in the front end, but some are understood in the front end and some are understood in the back end?

Does it provide the same pragma support across all code generators?
Are there conditional compilation capabilities?

What is the extent of the target features which are supported:
- 1750A (timer a, timer b, extended memory, etc.)
- CISC (whatever particular features are identified by the chip)
- RISC (whatever particular features are identified by the chip)

Is there an intelligent, modifiable linker?

Analysis

Is there a host-based target emulator, simulator, and symbolic debugger?
Is there a facility for supporting interoperability (communications paths) between simulated target processors for multi-target debugging?
Does the host development system have visibility into actual target processor hardware during execution? If so:
- Is such visibility in terms of original source code names?
- Is such visibility extendable into multiple targets?

Is there a host-based static target timing analysis capability?

Figure 14.3-1 Cross Development System Support Questionnaire

14.4 APSE CUSTOMIZATION QUESTIONNAIRE

Purpose: Evaluation of the APSE's ability to be customized for a particular host and target environment, methodology, or application domain.

[@RM: Whole APSE Issues	3,	(@RM: Application Independence	6.4.3,
		@RM: Augmentability	6.4.4,
		@RM: Commonality	6.4.7,
		@RM: Distributedness	6.4.12,
		@RM: Generality	6.4.16,
		@RM: Modularity	6.4.20,
		@RM: Operability	6.4.21,
		@RM: Rehostability	6.4.26,
		@RM: Required Configuration	6.4.27,
		@RM: Retargetability	6.4.28,
		@RM: System Compatibility	6.4.35);
@RM: Runtime Environment	7.2.3.5,	(@RM: Augmentability	6.4.4,
		@RM: Modularity	6.4.20)]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire.

Inputs: Questionnaire (see Fig. 14.4-1) and APSE documentation.

Process: Answer questions based on documentation, using the tools, or asking the vendor.

Outputs: Completed questionnaire.

Methodology

Can the user tailor the methodology supported to fit his own needs, such as for rapid prototyping or partial life cycle completion?

Can the user define his own methodology?

Automation

Is there development automation present in the APSE?

If so, can it be modified to reflect:

- different project management organization?
- different life cycle definition (standard or user-defined)?
- different document standards generation?
- different project-specific configuration management?

Documentation

Is there documentation support for:

- user-defined document formats?
- customer-defined document formats?

Can the configuration management for documentation be altered?

Does the APSE support the planning, design, generation, baseline, and maintenance of documents?

Is the information created by the APSE directly importable into the documents?

Can information from one document be transferred to another?

Are changes to the system automatically reflected in the system documentation?

Does the APSE support merged text and graphics documentation?

Is the documentation resident in one database or is it derived from multiple databases?

Is the documentation exportable to another APSE or another database?

User Role Change

Can the APSE be modified to support a user in:

- a different skill level?
- a different job assignment?

Communication

What communication protocols are supported by the APSE?

- Can they be modified?
- Can they be user-defined?
- Is special hardware required to support this communication?

Distribution

Can the APSE go from a single host to support multiple homogenous hosts? Heterogenous hosts?

Can the APSE go from an homogenous to an heterogenous environment and vice versa?

Host Dependencies

Can the APSE be modified for use on another host?

Is the APSE built on top of a portability interface implementation such as the CAIS?

Target Dependencies

Does the APSE support one target?

- Can it be modified?
- Can other targets be supported?

Can multiple targets be supported at one time or only one target?

Does the APSE support real-time embedded or non real-time, embedded targets only?

Runtime Support System

Can the RTS be modified?

Are there standard modifications (versions) provided?

Does it have a modular construction?

Is the design documentation for it provided?

- Is it easily understood?

Is the associated toolset (linker, loader, compiler) modifiable to support the modifications of the RTS?

Is the RTS source code provided?

Does the RTS support multiple targets?

Figure 14.4-1 APSE Customization Questionnaire

14.5 Ada-EUROPE: PROGRAM INTERACTION QUESTIONNAIRE

Purpose: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@Buxton 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided. Chapter 8 discusses issues associated with interaction between programs (program invocation, program communication, and kernel facilities for debugging).

[@RM: Whole APSE Issues	3,	(@RM: Augmentability	6.4.4,
		@RM: Commonality	6.4.7,
		@RM: System Compatibility	6.4.35)]

Primary References: [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986, Chapter 8.

[Ada-Europe: Selecting an Ada Environment 4.9]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire.

Inputs: Questionnaire, APSE, and APSE documentation.

Process: Answer the questions by using the APSE, reading the documentation, or asking the vendor of the APSE.

Outputs: Completed questionnaire.

15. INFORMATION MANAGEMENT SUPPORT ASSESSORS

These assessors examine the performance, usability, and completeness of the APSE or APSE component functionality related to controlling the information flow during the development of a software system. This includes the organization, accession, modification, dissemination, and processing of any associated information.

15.1 FILE MANAGEMENT CHECKLIST

Purpose: Evaluation of the completeness and power of file management by developing a list of functional capabilities.

[@RM: File Management	7.2.1.3,	(@RM: Completeness	6.4.9,
		@RM: Power	6.4.22)]

Primary References: [Peterson 1985] J.L. Peterson and A. Silberschatz, "Operating System Concepts," 2nd edition, Addison-Wesley, 1985.

Vendors/Agents: [E&V Team]

Method: Capabilities checklist.

Inputs: Capabilities checklist (see Table 15.1-1) and file manager documentation.

Process: Check off capabilities demonstrated by the file manager or discussed in the documentation.

Outputs: A list of capabilities provided by the file manager.

Table 15.1-1 File Management Capabilities Checklist

FEATURE	FOUND	NOTES
Operations		
Create		
Read		
Write		
Delete		
Rewind		
Append		
Copy		
Rename		
Update		
Compress		
Expand		
Compare		
Directories		
Operations		
Search		
Create Directory		
Delete Directory		
Rename Directory		
List Directory		
Backup		
Restore		
Structure		
Single-Level (Flat)		
Two-Level		
Tree-Structured (Hierarchical)		
Acyclic Graph		
General Graph		

Table 15.1-1 File Management Capabilities Checklist (Continued)

FEATURE	FOUND	NOTES
Storage Format Record Types Fixed Length Variable Length Byte Count at Beginning End of Record Marker Blocked Records Spanned Records Polymorphic Records Data Text ASCII EBCDIC Numbers Integers Signed Magnitude 1's Complement 2's Complement Floating Point IEEE Format Persistent Knowledge of Ada Types Media Disk Drum Magnetic Tape Other Multi-Volume Files Allocation Method Contiguous Linked Indexed		
Access Management Sequential File Direct (Random) Access File Primary Indexing Secondary Indexing Hash-Coded Indexing		
Access Security Protection Dynamic Protection Structure Data Encryption File Password Static Dynamic Multiple Access Control Controlled Operations Read Write Execute Append Delete Access Matrix Global Table Access List Capability List Lock/Key		

15.2 DATABASE MANAGEMENT CHECKLIST

Purpose: Evaluation of the completeness and power of database management by developing a list of functional capabilities.

[@RM: Database Management 7.2.1.1, (@RM: Completeness 6.4.9, @RM: Power 6.4.22)]

Primary References: [Martin 1986] D. Martin, "Advanced Database Techniques," MIT Press, 1986.

Vendors/Agents: [E&V Team]

Method: Capabilities checklist.

Inputs: Capabilities checklist (see Table 15.2-1) and database manager documentation.

Process: Check off capabilities demonstrated by the database manager or discussed in the documentation.

Outputs: A list of capabilities provided by the database manager.

Table 15.2-1 Database Management Capabilities Checklist

FEATURE	FOUND	NOTES
Model Hierarchical Network Relational Object-Oriented Other		
Data Dictionary Management Definitions Files, Tables Fields, Attributes Relationships Subschemas Views Single-Level Multilevel Objects, Entities Data Types Subtypes User-Defined Types Operations Implementation Parameters Non-Database Entities Listing Descriptions Cross-Referencing Descriptions Dictionary History Automatic Generation of Data Definition Statements		
Data Queries Non-Procedural Language (4GL) Fill-in-the-Form Query by Example		
Report Generation Query Data Set User-Defined		
Update Mode Static Dynamic		
Access Security Protection Access Control By View By File, Table By Object, Entity By Relationship By Operation (Read, Write, Append) Data Encryption Dynamic Password		
Keyword Input Protection		
Protection of Stored and Transmitted Data Referential Integrity		
Access Conflict and Deadlock Protection Access Locking Dynamic Backout from Deadlock Undoing Multiple Transactions		

Table 15.2-1 Database Management Capabilities Checklist (Continued)

FEATURE	FOUND	NOTES
Storage Full-Length Representation with Codes Data Packing		
Disk Space Management Multi-Volume Files Areas File Groups		
File Access Management Sequential File Direct (Random) Access File Primary Indexing Secondary Indexing Hashing Hash-Coded Index Database-Key Bit-Vector Inverted File (Bit-index)		
Entity Linking One-to-One Relationship One-to-Many Relationship Many-to-Many Relationship		
Application Program Interface Application Development Language Interface Standard DBMS Operations Insertion (Record Creation or Addition) Modification (Field Update) Deletion Link Creation and Suppression Screen Generators		
Program/Data Independence through Mapping		
Program/Structure Independence Using Multilevel Views		
Backup and Recovery Transaction Logging Cold Restart Warm Restart		
Data Restructuring Capabilities (Views)		
Administration Capabilities Interactive Dictionary Management Access Permission Management Data Quality Verification		
Communication Capabilities Import, Bulk Data Loading Export, Flat File Conversion Single System Access Multiple System Access Distributed Database		
Performance Monitoring/Tuning		
Miscellaneous Terminal Independent On-Line Help Facility		

15.3 ELECTRONIC MAIL CHECKLIST

Purpose: Evaluation of the completeness and power of electronic mail by developing a list of functional capabilities.

[@RM: Electronic Mail	7.2.1.4,	(@RM: Completeness	6.4.9,
		@RM: Power	6.4.22)]

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 15.3-1) and mail system documentation.

Process: Check off capabilities demonstrated by the mail system or discussed in the documentation.

Outputs: A list of capabilities provided by the mail system.

Table 15.3-1 Electronic Mail Capabilities Checklist

FEATURE	FOUND	NOTES
Notification of New Mail		
Receive Messages		
Read Messages Next Previous Show Message History/Status First Last Specified		
Edit Message Create Message Modify Header Copy Message Convert Message to Alternate Format Attach/Detach Documents		
Send Message to Distribution List across Network to cc: (carbon copy) List Registered Mail Defer Delivery Cancel Deferred Delivery		
Save Message File/Refile/Cross-File Message Create Folders Copy Folders Delete Folders Refile Folders Cross-File Folders Select Folders Reorganize File Cabinet Save/File Attachments		

Table 15.3-1 Electronic Mail Capabilities Checklist (Continued)

FEATURE	FOUND	NOTES
Print Message Single Multiple List/Print Messages/Folder Index		
Delete Message Single Multiple		
Miscellaneous Initialize Environment Configure Mailbox/Customize per User Preferences Manage Aliases Manage Distribution Lists Mark Messages/Unmark Messages/Commit Changes (for Deleting/Filing/Copying) Auto Forward/Cancel Auto Forward Auto Reply/Cancel Auto Reply Sort Messages Locate Message/Select by Content/Search for String Interfaces to Bulletin Board to Programs to Fax On-Line Help Facilities Keypad Support Mouse Support		

15.4 TRADE JOURNAL COMMUNICATIONS TOOL EVALUATIONS

Purpose: Various trade journals have evaluated and compared communications tools. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of communications tools, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: Electronic Mail	7.2.1.4,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Cost	6.4.11,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
(@RM: Text Editing	7.1.1.1,		
@RM: Pre & User-Defined Forms	7.1.2.3,		
@RM: Documentation Management	7.2.1.2,		
@RM: Spelling Checking	7.3.1.2),	@RM: Completeness	6.4.9;
@RM: Input/Output Support	7.2.3.2,	(@RM: Completeness	6.4.9,
		@RM: System Compatibility	6.4.35);
@RM: Emulation	7.3.2.13,	(@RM: Completeness	6.4.9,
		@RM: System Compatibility	6.4.35)]

Primary References: [Eva 1990] E. Eva, et al., Product Comparison, "Breaking Down Communications Barriers with E-Mail," *InfoWorld*, vol. 12, issue 23, 4 June 1990, pp. 83-109.

[Honan 1990] P. Honan and J. Desposito, Buyer's Guide, "Communications Software," *Personal Computing*, vol. 14, no. 4, 27 April 1990, pp. 109-139.

[Campbell 1989] G. Campbell, "Communications Software: Easy Choices," *PC World*, vol. 7, no. 8, August 1989, pp. 118-133.

[Simone 1989] L. Simone, "E-Mail, the Global Handshake," *PC Magazine*, vol. 8, no. 14, August 1989, pp. 175-210.

Vendors/Agents: [InfoWorld, Personal Computing, PC World, PC Magazine]

Method: Capabilities checklist(s) and performance benchmark tests.

Inputs: Checklists, benchmark tests, communications tool(s), and documentation.

Process: 1) Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.

2) Run the performance benchmark tests.

Outputs: Completed checklist(s) and performance reports.

15.5 TRADE JOURNAL DATABASE MANAGER EVALUATIONS

Purpose: Various trade journals have evaluated and compared database managers. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of database managers, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: Database Management	7.2.1.1,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
(@RM: Text Editing	7.1.1.1,		
@RM: Pre & User-Defined Forms	7.1.2.3,		
@RM: Sort/Merge	7.1.4,		
@RM: Encryption	7.1.6.18,		
@RM: Performance Monitoring	7.2.1.10,		
@RM: Math/Statistics	7.2.3.4,		
@RM: Debugging	7.3.2.5),	@RM: Completeness	6.4.9]

Primary References: [Petreley 1990] N. Petreley, Z. Banapour, and L. Slovik, Product Comparison, "Dueling Servers," *InfoWorld*, vol. 12, issue 10, 5 March 1990, pp. 57-75.

[Lee 1989] C. Lee, Buyer's Guide, "Database Managers," *Personal Computing*, vol. 13, no. 12, December 1989, pp. 151-180.

[Shaw 1989] R.H. Shaw, "Databases for OS/2: The First Wave," *PC Magazine*, vol. 8, no. 11, 13 June 1989, pp. 94-138.

Vendors/Agents: [InfoWorld, Personal Computing, PC Magazine]

Method: Capabilities checklist(s) and performance benchmark tests.

Inputs: Checklists, benchmark tests, database manager(s), and documentation.

Process: 1) Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.

2) Run the performance benchmark tests.

Outputs: Completed checklist(s) and performance reports.

16. OTHER ASSESSORS

This chapter contains instances of E&V technology that do not conveniently fit one of the earlier chapters. It is likely that in future versions of the Guidebook some of these "miscellaneous" instances will be grouped together in new chapters, and therefore moved out of Chapter 16.

16.1 TEXT EDITING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness and power of text editing by developing a list of functional capabilities.

[@RM: Text Editing 7.1.1.1, (@RM: Completeness 6.4.9, @RM: Power 6.4.22)]

Primary References: [E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[Classification Schema/E&V Taxonomy Checklists 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist.

Inputs: Capabilities checklist (see Table 16.1-1) and text editor documentation.

Process: Check off capabilities demonstrated during editing sessions or discussed in the documentation.

Outputs: A list of capabilities provided by the text editor.

Table 16.1-1 Text Editing Capabilities Checklist

FEATURE	FOUND	NOTES
Locator Movement Left, Right, Up, Down Top, Bottom of File Next/Previous Word Beginning, End of Line Beginning, End of Page (Screen) Scroll Up, Down, Left, Right Page Up, Down, Left, Right		
Search/Replace Search Forward, Backward Regular Expression Search, Replace Multiple Replace		
Buffers Copy Text To, From Edit Multiple Files Split Screen		
Regions Set Mark Insert Region Delete Region Copy Region Move Region Hide, Show Region		
File Manipulation Copy From File Append To File		
Macros Keyboard Macros Macro Language		
File Storage Save (Continue Editing) Quit (No Save) Automatic Save Versioning (Backup Original, Save Changes Only) Baselining		
Miscellaneous Terminal Independent On-Line Help Facility Minimal Redisplay Algorithm (Refresh) Key Redefinition Undo Command Command Recall, Redo Command Type-Ahead Session Logging Spawn Command Language Process		

16.2 LANGUAGE-SENSITIVE EDITING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness and power of language-sensitive editing by developing a list of functional capabilities. This list deals only with those features that provide the language-sensitivity to the editor. For a list of features supporting general text editing see the Text Editing Capabilities Checklist [16.1]. This list may be used to evaluate editors which are sensitive to languages such as Ada or FORTRAN as well as word processors which may be viewed as editors which are sensitive to the English language.

[@RM: Text Editing	7.1.1.1,	(@RM: Completeness @RM: Power	6.4.9, 6.4.22);
@RM: Syntax & Semantics Check	7.3.1.15,	(@RM: Completeness @RM: Power	6.4.9, 6.4.22)]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 16.2-1) and text editor documentation.

Process: Check off capabilities demonstrated during editing sessions or discussed in the documentation.

Outputs: A list of capabilities provided by the text editor.

Table 16.2-1 Language-Sensitive Editing Capabilities Checklist

FEATURE	FOUND	NOTES
Locator Movement Next, Previous Word (Identifier, Keyword) Beginning, End of Sentence (Statement, Comment) Beginning, End of Paragraph (Block) Beginning, End of Section (Unit) Beginning, End of Document (Compilation)		
Search/Replace Word - Where Used, Where Defined Sentence Paragraph Template Stub Section		
Regions Define Region - Word, Sentence, Paragraph, Template, Stub, Section Insert Region Delete Region Copy Region Move Region Hide, Show Region Comment Out Region		
Display High-Level Structure Unclosed Structures Matching Structures Permitted Constructs Words of Permitted Type		
Miscellaneous (User-Defined) Reformat On-Line LRM Access Support for Mixed Languages Analyze Change - Section, Document Check Spelling Check Grammar (Syntax) Check Meaning (Semantics) Translate (Compile) - Sentence, Paragraph, Section Knowledge-based versus Template-based Traceability between Objects		

16.3 PERFORMANCE MONITORING CHECKLIST

Purpose: Evaluation of the completeness of performance monitoring by developing a list of functional capabilities.

[@RM: Performance Monitoring 7.2.1.10, @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 16.3-1) and performance monitor documentation.

Process: Check off capabilities demonstrated by the performance monitor or discussed in the documentation.

Outputs: A list of capabilities provided by the performance monitor.

Table 16.3-1 Performance Monitor Capabilities Checklist

FEATURE	FOUND	NOTES
Hardware CPU Time (Real And Virtual) Memory Usage I/O Channel Traffic Terminal Response Terminal Connect Time Terminal Availability Disk Usage Disk Space Availability Tape Mounts Tape Drive Availability Printout Quantity		
Software Tool Usage Program Library Monitoring Wall Clock Time		

16.4 SCHEDULING CHECKLIST

Purpose: Assess the ability of the project management tools to depict the project milestones and their relationships in a timed schedule format.

[@RM: Scheduling 7.2.2.3; @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Checklist.

Inputs: Capabilities Checklist (see Table 16.4-1) and project management documentation.

Process: Check off capabilities cited in documentation and test the ones of interest to the project.

Outputs: A list of capabilities provided by the project management tools.

Table 16.4-1 Scheduling Checklist

FEATURE	FOUND	NOTES
CPM Graph Automatic generation of CPM from input data Automatic Gantt chart generation from the CPM		
PERT Chart Automatic generation of PERT from input data Automatic Gantt chart generation from the PERT		
Gantt Chart Depict duration required for project completion based on: Individual activity Group of activities Total project Milestone achievement Calendar time schedule Allow the assignment of resources to Gantt chart entities Allow the assignment of budget to Gantt chart entities		
Import/Export of Milestone Data between Projects Individual activity A group of activities Groups of activities Complete project Network of projects Milestones Resources Calendar time segments		
Integration/Merging of Data from Several Projects Individual activity A group of activities Groups of activities Complete project Network of projects Milestones Resources Calendar time segments		
Reporting Text, graphics and merged text and graphics Internal (user-defined format and contents) reports External (customer-defined format and contents) reports		

16.5 TRACKING CHECKLIST

Purpose: Assess the ability of the project management tools to periodically or continuously collect, analyze, and report current milestone and resource data to make project progress visible.

[@RM: Tracking 7.2.2.6; @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Checklist.

Inputs: Capabilities Checklist (see Table 16.5-1) and project management documentation.

Process: Check off capabilities cited in documentation and test the ones of interest to the project.

Outputs: A list of capabilities provided by the project management tools.

Table 16.5-1 Tracking Checklist

FEATURE	FOUND	NOTES
Resource Status Data		
Tracking of budget data collected and monitored by:		
Work Breakdown Structure		
Cost Element Structure		
Individual Activity		
Group of Activities		
Groups of Activities		
Complete Project		
Network of Projects		
Calendar Time Segments		
Loading to view resource loading on the basis of:		
Individual Activity		
Group of Activities		
Project Completion		
Milestone Achievement		
Calendar Time Schedule		
Percentage of Resource		
Priority of Resource Usage		
Directly from Gantt chart entities		
Directly from PERT chart entities		
Directly from CPM chart entities		
Assignment to track resource responsibilities on the basis of:		
Individual Activity		
Group of Activities		
Project Completion		
Milestone Achievement		
Calendar Time Schedule		
Usage definition to track resources in terms of:		
How it is used		
Where it is used		
By whom it is used		
Who is responsible for its use		
What percentage of it is used		
When that percentage is used		
What percentage is available for use		
When that percentage is available for use		
Identification of backup resources in the case of failure		
Cost of use		
Analysis to view the requirements changes and percentage of a resource exhausted on the following basis:		
Individual Activity		
Group of Activities		
Milestone		
Group of Milestones		
Calendar Time Schedule		

Table 16.5-1 Tracking Checklist (Continued)

FEATURE	FOUND	NOTES
Cost Management Tracking and comparison of budget data collected and monitored by: Work Breakdown Structure Cost Element Structure Individual Activity Group of Activities Groups of Activities Complete Project Network of Projects Calendar Time Segments Analysis to determine current earned value of an activity or resource to find: Percent of a task's duration that is completed Percent of a resource that has been used Consequences of changing budgetary allocations		
Milestone Status Data Tracking and comparison of milestone status data collected and monitored by: Work Breakdown Structure Cost Element Structure Individual Activity Group of Activities Groups of Activities Complete Project Network of Projects Calendar Time Segments Analysis to determine milestone requirements changes and percent completion for resources expended on the following basis: Individual Activity Group of Activities Milestone Group of Milestones Calendar Time Schedule		
Import/Export of Data between Projects Budget data Milestone status data Resource data		
Integration/Merging of Data from Several Projects Budget data Milestone status data Resource data		
Reporting Cost management reports Milestone status reports Resource status reports Ad hoc reports		

16.6 STEM/TRW DOCUMENTATION TOOLS EVALUATION

Purpose: Evaluation of the capabilities of documentation tools. The goals of this project are to develop evaluation procedures, identify and classify documentation tools, evaluate some tools, and contribute to the STEM database [@STSC 1990]. A hierarchical checklist has been developed, including the following top-level categories:

User Interface
Input Devices, Formats and Filters
Output Devices, Formats and Filters
Create/Edit Text, Tables, Graphics
Document Design, Composition
File/Database Management
Communications and Networking
System and Job Control
Hardware
Unique User Requirements

The cited reference provides second-level entries under all of the above categories.

[@RM: Text Editing	7.1.1.1,	(@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35);
@RM: Graphics Editing	7.1.1.3,	(@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35);
@RM: Formatting	7.1.2,	(@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35);
@RM: Documentation Management	7.2.1.2,	(@RM: Commonality	6.4.7,
		@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35)]

Primary References: [Satterthwaite 1990] L.H.Satterthwaite, "Documentation Tool Evaluation Project," USAF STSC - HQ USAF/SC Joint Software Conference Proceedings, Salt Lake City, 23-26 April 1990 [@STSC 1990].

[Software Tool Evaluation Model (STEM) 4.20]

Vendors/Agents: [STSC, TRW]

Method: Capabilities Checklist

Inputs: Capabilities checklist and documentation tool and its documentation.

Process: Check off capabilities demonstrated by testing or discussed in the documentation.

Outputs: A list of capabilities provided by the tool.

16.7 Ada-EUROPE: PROJECT MANAGEMENT QUESTIONNAIRE

Purpose: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@Buxton 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided. Chapter 16 discusses issues associated with project and product management (integration with a database, estimation planning and monitoring tools, quality management and configuration management, and social and legal aspects).

[(@RM: Cost Estimation	7.2.2.1,		
@RM: Quality Specification	7.2.2.2,		
@RM: Resource Estimation	7.2.2.5,		
@RM: Tracking	7.2.2.6),	@RM: Completeness	6.4.9]

Primary References: [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986, Chapter 16.

[Ada-Europe: Selecting an Ada Environment 4.9]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire.

Inputs: Questionnaire, tool, and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed questionnaire.

16.8 TRADE JOURNAL WORD PROCESSOR EVALUATIONS

Purpose: Various trade journals have evaluated and compared word processors. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of word processors, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth. The line between word processors, which traditionally focused on text processing, and desktop publishers, which traditionally focused on graphics and page layout, is becoming increasingly blurred (see also [16.9]).

[@RM: Text Editing	7.1.1.1,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Graphics Editing	7.1.1.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Formatting	7.1.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38);
@RM: Spelling Checking	7.3.1.2,	(@RM: Completeness	6.4.9,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27);
@RM: Syntax & Semantics Check	7.3.1.15,	(@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38)]

Primary References: [Honan 1990a] P. Honan and J. Devlin, Buyer's Guide, "Word Processing," *Personal Computing*, vol. 14, no. 8, August 1990, pp. 133-150.

[Campbell 1990] G. Campbell, "Picture-Perfect Word Processing," *PC World*, vol. 8, no. 5, May 1990, pp. 104-111.

[Lombardi 1990] J. Lombardi, Product Comparison, "Treasures Abound," *InfoWorld*, vol. 12, issue 5, 29 January 1990, pp. 89-115.

[Mendelson 1989a] E. Mendelson, "Two Aces and a King: The Big Three Word Processors Raise the Ante," *PC Magazine*, vol. 8, no. 20, 28 November 1989, pp. 97-122.

Vendors/Agents: [Personal Computing, PC World, InfoWorld, PC Magazine]

Method: Capabilities checklist(s) and performance benchmark tests.

Inputs: Checklists, benchmark tests, word processor(s), and documentation.

Process: 1) Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.

2) Run the performance benchmark tests.

Outputs: Completed checklist(s) and performance reports.

16.9 TRADE JOURNAL DESKTOP PUBLISHING EVALUATIONS

Purpose: Various trade journals have evaluated and compared desktop publishers. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of desktop publishers, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth. The line between desktop publishers, which traditionally focused on graphics and page layout, and word processors, which traditionally focused on text processing, is becoming increasingly blurred (see also [16.8]).

[@RM: Text Editing	7.1.1.1,	(@RM: Anomaly Management 6.4.2, @RM: Completeness 6.4.9, @RM: Documentation Quality 6.4.13, @RM: Operability 6.4.21, @RM: Power 6.4.22, @RM: Processing Effectiveness 6.4.23, @RM: Required Configuration 6.4.27, @RM: Storage Effectiveness 6.4.32, @RM: System Compatibility 6.4.35, @RM: Vendor Support 6.4.38);
@RM: Graphics Editing	7.1.1.3,	(@RM: Anomaly Management 6.4.2, @RM: Completeness 6.4.9, @RM: Documentation Quality 6.4.13, @RM: Operability 6.4.21, @RM: Power 6.4.22, @RM: Required Configuration 6.4.27, @RM: Storage Effectiveness 6.4.32, @RM: System Compatibility 6.4.35, @RM: Vendor Support 6.4.38);
@RM: Formatting	7.1.2,	(@RM: Anomaly Management 6.4.2, @RM: Completeness 6.4.9, @RM: Documentation Quality 6.4.13, @RM: Operability 6.4.21, @RM: Power 6.4.22, @RM: Processing Effectiveness 6.4.23, @RM: Required Configuration 6.4.27, @RM: Vendor Support 6.4.38);
@RM: Syntax & Semantics Check	7.3.1.15,	(@RM: Completeness 6.4.9, @RM: Documentation Quality 6.4.13, @RM: Operability 6.4.21, @RM: Power 6.4.22, @RM: Required Configuration 6.4.27, @RM: Vendor Support 6.4.38)]

Primary References: [Simone 1990] L. Simone, "Self-Sufficient Publishing," *PC Magazine*, vol. 9, no. 4, 27 February 1990, pp. 97-176.

[Assadi 1990] B. Assadi and G. Gruman, Product Comparison, "Putting the Best to the Test," *InfoWorld*, vol. 12, issue 1, 1 January 1990, pp. 39-55.

[Bell 1989] J. Bell and M. Young, Buyer's Guide, "Desktop Publishing Software," *Personal Computing*, vol. 13, no. 6, June 1989, pp. 115-170.

Vendors/Agents: [PC Magazine, InfoWorld, Personal Computing]

Method: Capabilities checklist(s) and performance benchmark tests.
Inputs: Checklists, benchmark tests, desktop publisher(s), and documentation.
Process: 1) Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.
2) Run the performance benchmark tests.
Outputs: Completed checklist(s) and performance reports.

16.10 TRADE JOURNAL PRESENTATION GRAPHICS EVALUATIONS

Purpose: Various trade journals have evaluated and compared presentation graphics packages. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of presentation graphics packages, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: Text Editing	7.1.1.1,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Data Editing	7.1.1.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Graphics Editing	7.1.1.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Pre & User-Defined Forms	7.1.2.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38);
@RM: Graphics Generation	7.1.5,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Math/Statistics	7.2.3.4,	(@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38)]

Primary References: [Fridlund 1990] A.J. Fridlund, Product Comparison, "Making Your Point - with Style," *InfoWorld*, vol. 12, issue 24, 11 June 1990, pp. 63-70.

[Devlin 1990] J. Devlin and J. Pepper, Buyer's Guide, "Business Graphics Software," *Personal Computing*, vol. 14, no. 2, February 1990, pp. 119-144.

[Jantz 1989] R. Jantz and M. Smith-Heimer, "Polished Presentations," *PC World*, vol. 7, no. 11, November 1989, pp. 116-131.

[Raskin 1989] R. Raskin, "The Packages Behind the Presentation," *PC Magazine*, vol. 8, no. 17, 17 October 1989, pp. 95-129.

Vendors/Agents: [InfoWorld, Personal Computing, PC World, PC Magazine]

Method: Capabilities checklist(s) and performance benchmark tests.
Inputs: Checklists, benchmark tests, presentation graphics package(s), and documentation.
Process: 1) Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.
2) Run the performance benchmark tests.
Outputs: Completed checklist(s) and performance reports.

16.11 TRADE JOURNAL SPREADSHEET EVALUATIONS

Purpose: Various trade journals have evaluated and compared spreadsheet packages. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of spreadsheet packages, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: Data Editing	7.1.1.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: Storage Effectiveness	6.4.32,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Pre & User-Defined Forms	7.1.2.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: Storage Effectiveness	6.4.32,
		@RM: Vendor Support	6.4.38);
@RM: Graphics Generation	7.1.5,	(@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38);
@RM: Sort/Merge	7.1.4,	(@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38);
@RM: Math/Statistics	7.2.3.4,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Required Configuration	6.4.27,
		@RM: Vendor Support	6.4.38)]

Primary References: [Stinson 1990] C. Stinson, "Spreadsheet Heavyweights Take on 1-2-3," *PC Magazine*, vol. 9, no. 8, 24 April 1990, pp. 97-156.

[Scoville 1990] R. Scoville, "Seven Sensible Spreadsheets," *PC World*, vol. 8, no. 4, April 1990, pp. 116-131.

[Apiki 1990] S. Apiki, et al., "Product Focus: Not Just for Numbers Anymore," *BYTE*, vol. 15, no. 2, February 1990, pp. 148-166.

[Walkenbach 1990] J. Walkenbach, Product Comparison, "High End Sheets," *InfoWorld*, vol. 12, issue 4, 22 January 1990, pp. 57-73.

[Hlavaty 1989] C. Hlavaty, Buyer's Guide, "Spreadsheet Software," *Personal Computing*, vol. 13, no. 11, November 1989, pp. 135-178.

Vendors/Agents: [PC Magazine, PC World, BYTE, InfoWorld, Personal Computing]

Method: Capabilities checklist(s) and performance benchmark tests.
Inputs: Checklists, benchmark tests, spreadsheet package(s), and documentation.
Process: 1) Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.
2) Run the performance benchmark tests.
Outputs: Completed checklist(s) and performance reports.

16.12 TRADE JOURNAL PROJECT MANAGEMENT EVALUATIONS

Purpose: Various trade journals have evaluated and compared project management packages. The value of the articles is not so much the assessment of the specific products (since they quickly become outdated), but the explanation of the how they perform the evaluation or comparison. Many articles go into some detail about what features should be in each of the products and how they evaluate performance and usability of the products. For somebody just starting an evaluation of project management packages, the articles are a good way to quickly come up to speed on what the issues are. In some cases the articles are a good way to perform a "first cut" evaluation to narrow the list of products to be evaluated in depth.

[@RM: Data Editing	7.1.1.2,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Documentation Management	7.2.1.2,	(@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: Vendor Support	6.4.38);
@RM: Electronic Mail	7.2.1.4,	(@RM: Anomaly Management	6.4.2,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38);
@RM: Scheduling	7.2.2.3,	(@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: System Accessibility	6.4.33,
		@RM: Vendor Support	6.4.38);
@RM: Work Breakdown Structure	7.2.2.4,	(@RM: Capacity	6.4.6,
		@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: Vendor Support	6.4.38);
@RM: Resource Estimation	7.2.2.5,	(@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: Processing Effectiveness	6.4.23,
		@RM: Vendor Support	6.4.38);
@RM: Tracking	7.2.2.6,	(@RM: Completeness	6.4.9,
		@RM: Documentation Quality	6.4.13,
		@RM: Operability	6.4.21,
		@RM: Power	6.4.22,
		@RM: System Accessibility	6.4.33,
		@RM: System Compatibility	6.4.35,
		@RM: Vendor Support	6.4.38)]

Primary References: [Lauriston 1990a] R.J. Lauriston, "Work-Group Software Worth Waiting For," *PC World*, vol. 8, no. 6, June 1990, pp. 122-138.

[Rupley 1989] S. Rupley, Product Comparison, "Meeting Makers," *InfoWorld*, vol. 11, issue 47, 20 November 1989, pp. 63-77.

[Derfler 1989] F.J. Derfler, "Imposing Efficiency: Workgroup Productivity Software," *PC Magazine*, vol. 8, no. 15, 26 September 1989, pp. 247-272.

[Heck 1989] M. Heck, Product Comparison, "Mission: Made Possible," *InfoWorld*, vol. 11, issue 39, 25 September 1989, pp. 57-76.

Vendors/Agents: [PC World, InfoWorld, PC Magazine]

Method: Capabilities checklist(s) and performance benchmark tests.

Inputs: Checklists, benchmark tests, project management package(s), and documentation.

Process: 1) Check off capabilities by using the tool, reading the documentation, or asking the vendor of the tool.
2) Run the performance benchmark tests.

Outputs: Completed checklist(s) and performance reports.

APPENDIX A
CITATIONS

[ACEC 1990] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Reader's Guide," Air Force Wright Research and Development Center, D500-12471-1, May 1990.

[ACEC 1990a] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) User's Guide," Air Force Wright Research and Development Center, D500-12470-1, May 1990.

[ACEC 1990b] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Version Description Document," Air Force Wright Research and Development Center, D500-12472-1, May 1990.

[ACVC 1989] Ada Compiler Validation Procedures, Version 2.0, AJPO, May 1989.

[Albus 1986] J.A. Albus, H.G. McCain, and R. Lumia, "NASA/NBS Reference Model for Telerobot Control System Architecture (NASREM)," National Bureau of Standards, Robot System Division, 4 December 1986.

[ANSI 1988] "Information Resource Dictionary System (IRDS)," American National Standard for Information Systems, ANSI X3.138-1988, American National Standards Institute, Inc., New York, NY, 1988.

[Apiki 1990] S. Apiki, et al., "Product Focus: Not Just for Numbers Anymore," *BYTE*, vol. 15, no. 2, February 1990, pp. 148-166.

[ARTEWG 1987] "Catalogue of Ada Runtime Implementation Dependencies," Association for Computing Machinery, Special Interest Group on Ada, Ada Runtime Environment Working Group, 1 December 1987.

[ARTEWG 1988] "A Framework for Describing Ada Runtime Environments," Proposed by Ada Runtime Environment Working Group (SIGAda), Ada Letters, Volume VIII, Number 3, May/June 1988, pp. 51-68.

[Assadi 1990] B. Assadi and G. Gruman, Product Comparison, "Putting the Best to the Test," *InfoWorld*, vol. 12, issue 1, 1 January 1990, pp. 39-55.

[**Baram 1989**] G. Baram and G. Steinberg, "Selection Criteria for Analysis and Design CASE Tools," Software Engineering Notes, ACM SIGSoft, Vol. 14, No. 6, pp. 73-80, October 1989.

[**Barnes 1985**] *Proceedings of the International Ada Conference, Paris*, eds. J.G.P. Barnes and G.A. Fisher, Jr, Cambridge University Press, 1985.

[**Barstow 1981**] D.R. Barstow and H.E. Shrobe, "Observations on Interactive Programming Environments," [**@Wasserman 1981**], pp. 286-301, 1981.

[**Bell 1989**] J. Bell and M. Young, Buyer's Guide, "Desktop Publishing Software," *Personal Computing*, vol. 13, no. 6, June 1989, pp. 115-170.

[**Berk 1990**] K.J. Berk and R.P. Hanrahan, "Evaluating Tools and Environments Concept," Proceedings of the IEEE 1990 National Aerospace and Electronics Conference, NAECON 1990, May 1990, Volume 2, pp. 658-663.

[**Bowen 1985**] T.P. Bowen, G.B. Wigle, and J.T. Tsai, "Specification of Software Quality Attributes Software Quality Evaluation Guidebook," Rome Air Development Center, Griffiss AFB, RADC-TR-85-37, Volume III (of three), February 1985, DTIC Number AD A153 990.

[**Brody 1989**] A. Brody, Product Comparison, "The Experts," *InfoWorld*, vol. 11, issue 25, 19 June 1989, pp. 59-75.

[**Brown 1990**] B. Brown, "Playing the DOS Shell Game," *PC Magazine*, vol. 9, no. 11, 12 June 1990, pp. 185-244.

[**Buxton 1980**] J.N. Buxton, "Requirements for Ada Programming Support Environments - STONEMAN," U.S. Department of Defense, February 1980, DTIC Number AD A100 404.

[**Byrne 1990**] D.J. Byrne and R.C. Ham, "Ada Versus FORTRAN Performance Analysis Using the ACPS," [**@PIWG 1990**], pp. 139-145, 1990.

[**Campbell 1989**] G. Campbell, "Communications Software: Easy Choices," *PC World*, vol. 7, no. 8, August 1989, pp. 118-133.

[**Campbell 1990**] G. Campbell, "Picture-Perfect Word Processing," *PC World*, vol. 8, no. 5, May 1990, pp. 104-111.

[**Castor 1984**] V.L. Castor, "Evaluation and Validation (E&V) Team Public Report," Volume I, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, 30 November 1984, DTIC Number AD A153 609.

[**CECOM 1989**] "Procedures For Computer-Aided Software Engineering Tool Assessment," CECOM Center for Software Engineering, US Army Communications-Electronics Command, ?, April 1989, DTIC Number pending.

[CECOM 1989a] "Evaluation Of Existing CASE Tools For Tactical Embedded Systems," CECOM Center for Software Engineering, US Army Communications-Electronics Command, ?, April 1989, DTIC Number pending.

[CIVC 1989] "CIVC Implementor's Guide," CIVC-FINL-19-1, SofTech, Inc., 16 October 1989.

[CIVC 1990] "CIVC1 Framework," CVC-VREL-2/1-01, SofTech, Inc., 1 March 1990.

[CIVC 1990a] "Test Report Reader's Guide with Appendix 1 - Operator's Guide," CIVC-FINL-020-02, SofTech, Inc., 19 March 1990.

[CIVC 1990b] "CIVC 2.0 Beta Test Suite Operator's Guide," CIVC-FINL-20-03, 18 July 1990.

[Clapp 1986] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, and T. Schultze, "Toward Real-Time Performance Benchmarks for Ada," Robotics Research Laboratory, Dept. of Electrical Engineering and Computer Science, The Univ. of Michigan, RSD-TR-12-86, July 1986.

[Clapp 1986a] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, and T. Schultze, "Toward Real-Time Performance Benchmarks for Ada," Communications of the ACM, Volume 29, Number 8, August 1986, pp. 760-778.

[Clapp 1990] R.M. Clapp and T. Mudge, "Parallel and Distributed Issues," [@PIWG 1990], pp. 33-37, 1990.

[DACS 1979] The DACS Glossary, A Bibliography of Software Engineering Terms, October 1979.

[DeMillo 1986] R.A. DeMillo, "Functional Capabilities of a Test and Evaluation Subenvironment in an Advanced Software Engineering Environment," Georgia Institute of Technology GIT-SERC-86/07, 20 October 1986.

[Derfler 1989] F.J. Derfler, "Imposing Efficiency: Workgroup Productivity Software," *PC Magazine*, vol. 8, no. 15, 26 September 1989, pp. 247-272.

[Devlin 1990] J. Devlin and J. Pepper, Buyer's Guide, "Business Graphics Software," *Personal Computing*, vol. 14, no. 2, February 1990, pp. 119-144.

[Dewan 1988] P. Dewan, "A Framework for Analyzing User Interfaces," Software Engineering Research Center, Purdue University, SERC-TR-11-P, 24 May 1988.

[DoD APSE Analysis 1984] [@E&V Report: DoD APSE Analysis Report C.]

[DoD 1977] DoD, "Requirements for High Order Computer Languages (IRONMAN)," U.S. Department of Defense, 1977, DTIC Number AD A100 403.

[**DoD 1982**] "Software Development Methodologies and Ada (METHODMAN)," U.S. Department of Defense, 1982.

[**DoD 1983**] ANSI/MIL-STD-1815A-1983, Reference Manual for the Ada Programming Language, U.S. Department of Defense, 17 February 1983, DTIC Number AD A131 511.

[**DoD 1983a**] "Trusted Computer System Evaluation Criteria," CSC-STD-001-83, U.S. Department of Defense Computer Security Center, 15 August 1983, DTIC Number AD A207 905.

[**DoD 1986**] DoD-STD-1838, Common APSE Interface Set (CAIS), U.S. Department of Defense, 9 October 1986, DTIC Number AD A157 589.

[**DoD 1989**] "Common APSE Interface Set, Revision A," MIL-STD-1838A, U.S. Department of Defense, April 1989, DTIC Number AD A157 589.

[**Donaldson 1988**] C. Donaldson and P.B. Dyson, "Computer-Aided Systems and Software Engineering Products for Time-Critical Applications Development," Software Productivity Solutions (SPS), Inc., April 1988.

[**Donohoe 1990**] P. Donohoe, R. Shapiro, and N.H. Weiderman, "Hartstone Benchmark User's Guide, Version 1.0," SEI-90-UG-1, Software Engineering Institute, March 1990.

[**Donohoe 1990a**] P. Donohoe, R. Shapiro, and N.H. Weiderman, "Hartstone Benchmark Results and Analysis, Version 1.0," SEI-90-TR-7, Software Engineering Institute, June 1990.

[**E&V Plan**] **[@Castor 1984: E&V Plan A]. [@Szymanski 1987: E&V Plan A].**

[**E&V Reference Manual**] **[@RM].**

[**E&V Requirements 1984**] **[@Castor 1984: E&V Requirements B.].**

[**E&V Requirements 1987**] **[@Szymanski 1987: E&V Requirements D].**

[**E&V Schema 1987**] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[**E&V Tools and Aids 1990**] **[@Szymanski 1990: Tools and Aids H].**

[**Eva 1990**] E. Eva, et al., Product Comparison, "Breaking Down Communications Barriers with E-Mail," *InfoWorld*, vol. 12, issue 23, 4 June 1990, pp. 83-109.

[**Fersko-Weiss 1990**] H. Fersko-Weiss, "CASE Tools for Designing Your Applications," *PC Magazine*, vol. 9, no. 2, 30 January 1990, pp. 213-251.

[**FIPS 1989**] "Information Resource Dictionary System (IRDS)," Federal Information Processing Standard Publication, FIPS PUB 156, National Institute of Standards and Technology, Gaithersburg, MD, April 5, 1989.

[**Firth 1987**] R. Firth, V. Mosley, R. Pethia, L. Roberts, W. Wood, "A Guide to the Classification and Assessment of Software Engineering Tools," Software Engineering Institute, Technical Report, CMU/SEI-87-TR-10, August 1987, DTIC Number AD A182 895.

[**Fridlund 1990**] A.J. Fridlund, Product Comparison, "Making Your Point - with Style," *InfoWorld*, vol. 12, issue 24, 11 June 1990, pp. 63-70.

[**Goforth 1990**] A. Goforth, P. Collard, and M. Marquardt, "Performance Measurement of Parallel Ada: An Application Based Approach," [**@PIWG 1990**], pp. 38-58, 1990.

[**Goodwin 1990**] M. Goodwin, "Disk Trouble? No, Thanks," *PC World*, vol. 8, no. 7, July 1990, pp. 133-148.

[**Gray 1987**] L. Gray, "Using the SEI's Methodology for Evaluating Ada Environments: A Comparison of VAX/VMS to Rational," Proceedings of the AIAA Computers in Aerospace VI Conference, 7-9 October 1987.

[**Grund 1985**] E.C. Grund, L.A. Hilliard, and K.A. Younger, "Key Characteristics of Ada Programming Support Environments," MITRE Corporation, ESD-TR-85-144, 9590, July 1985, DTIC Number AD B096 137.

[**Hampton 1990**] J.Hampton, D.Dyer, and G.Daich, "STEM for Test Tools," [**@STSC 1990**], 1990.

[**Heck 1989**] M. Heck, Product Comparison, "Mission: Made Possible," *InfoWorld*, vol. 11, issue 39, 25 September 1989, pp. 57-76.

[**Henderson 1987**] P.B. Henderson and D. Notkin, "Integrated Design and Programming Environment," *Computer*, IEEE, November 1987.

[**Hlavaty 1989**] C. Hlavaty, Buyer's Guide, "Spreadsheet Software," *Personal Computing*, vol. 13, no. 11, November 1989, pp. 135-178.

[**Honan 1990**] P. Honan and J. Desposito, Buyer's Guide, "Communications Software," *Personal Computing*, vol. 14, no. 4, 27 April 1990, pp. 109-139.

[**Honan 1990a**] P. Honan and J. Devlin, Buyer's Guide, "Word Processing," *Personal Computing*, vol. 14, no. 8, August 1990, pp. 133-150.

[**Hook 1985**] A.A. Hook, G.A. Riccardi, M. Vilot, and S. Welke, "User's Manual for the Prototype Ada Compiler Evaluation Capability (ACEC)," Version 1, Institute for Defense Analysis, IDA Paper P-1879, October 1985, DTIC Number AD A163 272.

[**Houghton 1983**] R.C. Houghton, Jr., "A Taxonomy of Tool Features for the Ada Programming Support Environment (APSE)," U.S. Department of Commerce, National Bureau of Standards, NBSIR-81-2625, December 1982, Issued February 1983.

[**Houghton 1987**] R.C. Houghton, Jr. and D.R. Wallace, "Characteristics and Functions of Software Engineering Environments: An Overview," ACM Software Engineering Notes, Vol. 12, No. 1, January 1987.

[**Howden 1982**] W.E. Howden, "Contemporary Software Development Environments," Communications of the ACM 25(5), pp. 318-329, 1982.

[**ISTAR 1987**] Workshop on Future Development Environments, Information Science and Technology Assessment for Research, Conference on Information Mission Area (IMA) Productivity, Department of Army Director of Information Systems for Command, Control, Communications and Computers, 13-15 April 1987, pp 28.

[**Jackson 1985**] A.R. Jackson, "Abstract Data Types and the IPSE Database," [**@McDermid 1985**], pp. 135-145, 1985.

[**Jantz 1989**] R. Jantz and M. Smith-Heimer, "Polished Presentations," *PC World*, vol. 7, no. 11, November 1989, pp. 116-131.

[**Kean 1985**] E.S. Kean and F.S. Lamonica, "A Taxonomy Of Tool Features For A Life Cycle Software Engineering Environment," Rome Air Development Center, Griffiss AFB, June 1985, DTIC Number AD B096 355.

[**Lauriston 1990**] R. Lauriston, "Cache Values," *PC World*, vol. 8, no. 2, February 1990, pp. 130-139.

[**Lauriston 1990a**] R.J. Lauriston, "Work-Group Software Worth Waiting For," *PC World*, vol. 8, no. 6, June 1990, pp. 122-138.

[**Lauriston 1990b**] R. Lauriston, "Hard Disk Health Insurance," *PC World*, vol. 8, no. 7, July 1990, pp. 109-118.

[**Law 1988**] M.H. Law, "Guide to Information Resource Dictionary System Applications: General Concepts and Strategic Systems Planning," NBS Special Publication 500-152, National Bureau of Standards, Gaithersburg, MD, April 1988.

[**Lawlis 1989**] P.K. Lawlis, "Supporting Selection Decisions Based on the Technical Evaluation of Ada Environments and Their Components," PhD. dissertation, Arizona State University, August 1989.

[**Lee 1989**] C. Lee, Buyer's Guide, "Database Managers," *Personal Computing*, vol. 13, no. 12, December 1989, pp. 151-180.

[**Lehman 1981**] M.M. Lehman, "The Environment of Program Development, Maintenance Programming, and Program Support," [**@Wasserman 1981**], pp. 3-14, 1981.

[**Lombardi 1990**] J. Lombardi, Product Comparison, "Treasures Abound," *InfoWorld*, vol. 12, issue 5, 29 January 1990, pp. 89-115.

[**Long 1988**] F.W. Long, and M.D. Tedd, "Evaluating Tool Support Interfaces," *Ada in Industry*, Proceedings of the Ada-Europe Conference, Munich, 7-9 June 1988, Cambridge University Press, 1988.

[**Lyons 1986**] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986.

[**Marshall 1990**] P. Marshall, Product Comparison, "Giving DOS a New Face," *InfoWorld*, vol. 12, issue 16, 16 April 1990, pp. 57-77.

[**Martin 1986**] D. Martin, "Advanced Database Techniques," MIT Press, 1986.

[**McClure 1989**] C. McClure, "The CASE Experience," *BYTE*, vol. 14, no. 4, April 1989, pp. 235-245.

[**McDermid 1984**] J. McDermid and K. Ripken, "Life Cycle Support in the Ada Environment," Cambridge University Press, 1984.

[**Mendelson 1989**] E. Mendelson, "Backup Software: For the Moment After," *PC Magazine*, vol. 8, no. 8, August 1989, pp. 269-322.

[**Mendelson 1989a**] E. Mendelson, "Two Aces and a King: The Big Three Word Processors Raise the Ante," *PC Magazine*, vol. 8, no. 20, 28 November 1989, pp. 97-122.

[**Mendelson 1990**] E. Mendelson, "Disaster Relief: DOS Utilities Save the Day," *PC Magazine*, vol. 9, no. 6, 27 March 1990, pp. 97-132.

[**Nejmeh 1989**] B.A. Nejmeh, "Characteristics of Integrable Software Tools," Software Productivity Consortium, INTEG_S/W_TOOLS-89036-N, Version 1.0, 23 May 1989.

[**Nissen 1984**] J.C.D. Nissen, B.A. Wichmann, et al.,"Guidelines for Ada Compiler Specification and Selection," in *Ada: Language, Compilers And Bibliography*, ed. M.W. Rogers, Cambridge University Press, 1984.

[**Notkin 1981**] D.S. Notkin and A.N. Habermann, "Software Development Environment Issues as Related to Ada," [**@Wasserman 1981**], pp. 107-133, 1981.

[**Oberndorf 1988**] P.A. Oberndorf, "The Common Ada Programming Support Environment (APSE) Interface Set (CAIS)," IEEE Transactions on Software Engineering, Vol. 14, No. 6, June 1988.

[**Peterson 1985**] J.L. Peterson and A. Silberschatz, "Operating System Concepts," 2nd edition, Addison-Wesley, 1985.

[**Petreley 1990**] N. Petreley, Z. Banapour, and L. Slovik, Product Comparison, "Dueling Servers," *InfoWorld*, vol. 12, issue 10, 5 March 1990, pp. 57-75.

[**Petrick 1989**] B.A. Petrick and S.J. Yanke, "An Analysis of *The Definition of a Production Quality Ada Compiler*," Engineering Group, The Aerospace Corporation, Volume I (SSD-TR-89-81) and Volume II (SSD-TR-89-82, PQAC Test Suite), 13 March 1989.

[**Pierce 1986**] R.H. Pierce, I. Marshall, and S.D. Blude, "An Introduction to the MoD Ada Evaluation System," Software Sciences Ltd., Report Number 5485, June 1986.

[**PIWG 1990**] "Ada Performance Issues," Ada Letters, Special Edition from SIGAda, the ACM Special Interest Group on Ada Performance Issues Working Group, Vol. X, Number 3, Winter 1990.

[**Rainer 1986**] S.R. Rainer and T.P. Reagan, "User's Manual for the Ada Compilation Benchmark Generator Tool," MITRE Corp. MTR-87W00192-01, January 1988.

[**Raskin 1989**] R. Raskin, "The Packages Behind the Presentation," *PC Magazine*, vol. 8, no. 17, 17 October 1989, pp. 95-129.

[**RM**] "Evaluation and Validation (E&V) Reference Manual," Version 3.0, Wright Research and Development Center, WRDC TR-90-?, Wright Patterson AFB, October 1990, DTIC Number pending.

[**Roybal 1990**] R. Roybal, "AIM Procurement Guide," AIM Technology, Version 1.0, 1990.

[**Rupley 1989**] S. Rupley, Product Comparison, "Meeting Makers," *InfoWorld*, vol. 11, issue 47, 20 November 1989, pp. 63-77.

[**Satterthwaite 1990**] L.H. Satterthwaite, "Documentation Tool Evaluation Project," [**@STSC 1990**], 1990.

[**Scoville 1990**] R. Scoville, "Seven Sensible Spreadsheets," *PC World*, vol. 8, no. 4, April 1990, pp. 116-131.

[**Shaw 1989**] R.H. Shaw, "Databases for OS/2: The First Wave," *PC Magazine*, vol. 8, no. 11, 13 June 1989, pp. 94-138.

[**Simone 1989**] L. Simone, "E-Mail, the Global Handshake," *PC Magazine*, vol. 8, no. 14, August 1989, pp. 175-210.

[**Simone 1990**] L. Simone, "Self-Sufficient Publishing," *PC Magazine*, vol. 9, no. 4, 27 February 1990, pp. 97-176.

[**STARS 1985**] "Proposed Version 001.0," STARS Joint Service Team for Software Engineering Environments, Stars Joint Program Office, October 1985.

[**Stenning 1981**] V. Stenning, T. Froggart, R. Gilbert, and E. Thomas, "The Ada Environment: A Perspective," [**@Wasserman 1981**], pp. 36-46, 1981.

[**Stinson 1990**] C. Stinson, "Spreadsheet Heavyweights Take on 1-2-3," *PC Magazine*, vol. 9, no. 8, 24 April 1990, pp. 97-156.

[**STSC 1990**] "STSC Strategy," USAF STSC - HQ USAF/SC Joint Software Conference Proceedings, Salt Lake City, 23-26 April 1990.

[**Stuebing 1988**] H.G. Stuebing, "Evaluation of Computer Aided Systems/Software Engineering Products for Time-Critical Naval Systems," *Proceedings of the Conference Methodologies and Tools for Real Time Systems*, November 14-15, 1988.

[**Sullivan-Trainor 1990**] M.L. Sullivan-Trainor, Product Spotlight, "Buyer's Scorecard," *ComputerWorld*, vol. 24, no. 15, 9 April 1990, pp. 68-69.

[**Szymanski 1985**] R. Szymanski, "Evaluation and Validation (E&V) Team Public Report," Volume II, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, November 1985, DTIC Number AD A172 343.

[**Szymanski 1987**] R. Szymanski, "Evaluation and Validation (E&V) Team Public Report," Volume III, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, September 1987, DTIC Number AD A196 164.

[**Szymanski 1990**] R. Szymanski, "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, December 1990, DTIC Number to be assigned.

[**Texas Instruments 1985**] The APSE Interactive Monitor, Texas Instruments, Slide Presentation to the E&V Team, 5 September 1985.

[**Troy 1987**] D.A. Troy, "An Evaluation of CASE Tools," Proceedings of the 1987 International Computer Software and Applications Conference, IEEE Computer Society, pp. 124-130, 1987.

[**Van Buren 1990**] J.K. Van Buren, "Evaluation Procedures for Requirements Analysis and Design CASE Tools," [**@STSC 1990**], 1990.

[**Von Gerichten 1989**] L. Von Gerichten, et al., "Software Methodology Catalog," CECOM Center for Software Engineering, US Army Communications-Electronics Command, C01-091JB-0001-01, March 1989, DTIC Number AD A210 548.

[**Walkenbach 1990**] J. Walkenbach, Product Comparison, "High End Sheets," *InfoWorld*, vol. 12, issue 4, 22 January 1990, pp. 57-73.

[**Walkenbach 1990a**] J. Walkenbach, "No-Excuses Backup Software," *PC World*, vol. 8, no. 7, July 1990, pp. 149-164.

[**Wasserman 1981**] A.I. Wasserman, *Tutorial: Software Engineering Environments*, IEEE, 1981.

[**Weiderman 1987**] N.H. Weiderman and A.N. Habermann, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, DTIC Number AD A180 905.

[**Weiderman 1987a**] N.H. Weiderman, et al., "Ada for Embedded Systems: Issues and Questions," Software Engineering Institute, Technical Report CMU/SEI-87-TR-26, December 1987, DTIC Number AD A191 096.

[**Weiderman 1989**] N.H. Weiderman, "Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0," Software Engineering Institute, Technical Report CMU/SEI-89-TR-13, March 1989, DTIC Number AD A207 717.

[**Weiderman 1989a**] N.H. Weiderman, "Hartstone: Synthetic Benchmark Requirements for Hard Real-Time Applications," SEI-89-TR-23, Software Engineering Institute, June 1989. (reprinted in [**@PIWG 1990**])

APPENDIX B
ACRONYMS AND ABBREVIATIONS

4GL	Fourth Generation Language (e.g., SQL)
ACEC	Ada Compiler Evaluation Capability
ACM	Association for Computing Machinery
ACPS	Ada Compiler Performance Test Suite
ACSI	Advanced Computing Solutions, Inc.
ACVC	Ada Compiler Validation Capability
AES	Ada Evaluation System
AFB	Air Force Base
AFWAL	Air Force Wright Aeronautical Laboratories (now WL)
AIAA	American Institute of Aeronautics and Astronautics
AIE	Ada Integrated Environment
AJPO	Ada Joint Program Office
ALS	Ada Language System
ALS/N	Ada Language System/Navy
ANNA	Annotation Language for Ada
ANSI	American National Standards Institute
APSE	Ada Programming Support Environment
APSEWG	APSE Working Group (E&V Team)
ARTEWG	Ada RunTime Environment Working Group (SIGAda)
ASCII	American Standard Code for Information Interchange
ATF	Advanced Tactical Fighter
AVF	Ada Validation Facility
AVO	Ada Validation Organization
BGT	Benchmark Generator Tool
bpi	bits per inch
BSI	British Standards Institute (UK)
CACM	Communications of the ACM
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAIS	Common APSE Interface Set
CAM	Computer Aided Manufacturing
CASE	Computer Aided Software Engineering
CECOM	Communications-Electronics Command (US Army)
CISC	Complex Instruction Set Computer
CIVC	CAIS Implementation Validation Capability

CLI	Command Language Instruction
CMU	Carnegie Mellon University
COBOL	COmmon Business Oriented Language
CORE	Controlled Requirements Expression
CPM	Critical Path Method
CPU	Central Processing Unit
CSC	Computer Software Component
CSU	Computer Software Unit
DACS	Data and Analysis Center for Software
DBMS	DataBase Management System
DoD	Department of Defense
DOS	Disk Operating System
DTIC	Defense Technical Information Center
EBCDIC	Extended Binary Coded Decimal Interchange Code
ESD	Electronic Systems Division (USAF)
E&V	Evaluation and Validation
FIPS	Federal Information Processing Standard
FORTRAN	FORmula TRANslation (language)
GB	Guidebook
GIT	Georgia Institute of Technology
GKS	*Graphical Kernel System*
HQ	HeadQuarters
IBM	International Business Machines Corporation
IDA	Institute for Defense Analysis
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IMA	Information Mission Area
INSTEP	INnovative SofTware Engineering Practices, Inc.
IPSE	Integrated Project Support Environment
IRD	Information Resource Dictionary
IRDS	Information Resource Dictionary System
ISTAR	Information Science and Technology Assessment for Research
I/O	Input/Output
JIAWG	Joint Integrated Avionics Working Group
KAPSE	Kernel Ada Programming Support Environment
KIT	KAPSE Interface Team
KITIA	KAPSE Interface Team for Industry and Academia

LHX	Light Helicopter Experimental
LISP	LISt Processing (language)
LRM	Language Reference Manual [@DoD 1983]
MAPSE	Minimal Ada Programming Support Environment
MCCS	Mission-Critical Computer System
MIL	MILitary (DoD)
MIPS	Million Instructions Per Second
MIT	Massachusetts Institute of Technology
MMI	Man-Machine Interface
MoD	Ministry of Defense (UK)
NADC	Naval Air Development Center
NASA	National Air and Space Administration
NBS	National Bureau of Standards (now NIST)
NIST	National Institute of Standards and Technology (formerly NBS)
NTIS	National Technical Information Service
OCD	Operational Concept Document
OS	Operating System
PC	Personal Computer
PCTE	Portable Common Tool Environment
PDL	Program Design Language
PERT	Project Evaluation and Review Technique
PIWG	Performance Issues Working Group (SIGAda)
POSIX	Portable Operating System for computer environments (IEEE 1003.1)
PQAC	Production Quality Ada Compiler test suite
RADC	Rome Air Development Center
RAM	Random Access Memory
REQWG	REQuirements Working Group (E&V Team)
RISC	Reduced Instruction Set Computer
RM	Reference Manual
RTE	RunTime Environment
RTS	RunTime System
SAIC	Science Applications International Corporation
SDE	Software Development Environment
SDI	Strategic Defense Initiative
SEE	Software Engineering Environment
SEI	Software Engineering Institute
SERC	Software Engineering Research Center
SIGAda	Special Interest Group for Ada (ACM)
SIGSoft	Special Interest Group for Software Engineering (ACM)
SPC	Software Productivity Consortium
SPS	Software Productivity Solutions

SQL	Structured Query Language
STARS	Software Technology for Adaptable, Reliable Systems
STD	STanDard (DoD, MIL)
STEM	Software Tool Evaluation Model (STSC)
STSC	Software Technology Support Center
TASC	The Analytic Sciences Corporation
TCP/IP	Transmission Control Protocol/Internet Protocol
TOR	Technical Operating Report
TR	Technical Report
UK	United Kingdom
UNIX	UNiplexed Information and Computing Service (UNICS)
US	United States
USAF	US Air Force
VAX	Virtual Address Extension
VMS	Virtual Memory System
VSR	Validation Summary Report
V&V	Verification and Validation
WIS	WWMCCS Information System
WL	Wright Laboratories (formerly WRDC)
WRDC	Wright Research and Development Center (formerly AFWAL)
WWMCCS	WorldWide Military Command and Control System

APPENDIX C
FORMAL GRAMMAR

This appendix specifies sections of the Reference Manual and Guidebook (Reference System) as a formal grammar. The sections include chapters four through seven of the Reference Manual (RM), chapters four through 16 of the Guidebook (GB), all explicit references, the tables of contents, the indices, and the citations. The specification is presented as a partitioned grammar for convenience.

(The grammar is presented in a modified Backus-Naur form. Brackets represent optionality when alone, and may be marked by an asterisk "*" to denote 0-N instances of the production, or by a sharp "#" to denote 1-N instances. Angle brackets denote comments in place of productions which are too elaborate to express here. All terminals of the grammar are expressed as quoted literals, or composite literals based on characters and character strings.)

C.1 FORMAL REFERENCES

Throughout the Reference System, whenever formal references are made, a single consistent set of grammar rules are used. This includes reference from one volume to the other, reference from one section in a volume to another section in the same document, and reference to documents outside the Reference System.

```
reference_list      ::= "[" references [";" references ]* "]"

references          ::= ["("] reference [ "," reference ]* [")"]

reference           ::= ["@"] phrase [ ":" [ phrase ] [ designator_list ]]
                        | [ phrase ] designator_list
```

phrase_list ::= phrase ["," phrase]*

phrase ::= <text lacking special characters>

designator_list ::= designator ["," designator]*

designator ::= [lead "."] lead ["." digits]*

lead ::= [digits] caps

digits ::= one_to_nine [zero_to_nine]*

one_to_nine ::= ("1"-"9")

zero_to_nine ::= ("0"-"9")

caps ::= ("A"-"Z")

C.2 FORMAL CHAPTERS

The formal chapters of the Guidebook are defined here.

C.2.1 Chapter Components

The following rules define the components which are used to compose formal chapter entries.

prolog ::= header purpose primary [host]
[vendors_agents]

header ::= designator phrase

purpose ::= "Purpose:" text

primary ::= "Primary References:" reference_list

host ::= "Host/OS:" text

vendors_agents ::= "Vendors/Agents:" reference_list

method ::= meth_description inputs
process outputs

meth_description ::= "Method:" text

inputs ::= "Inputs:" text

process ::= "Process:" text

outputs ::= "Outputs:" text

citations ::= "Citations:" [citations]#

synopsis_text ::= "Synopsis:" text

methods ::= "Methods:" reference_list

text ::= < prose text >

C.2.2 Chapter Entries

Each numbered section of the formal chapters follows a specific grammar rule. The following rules define the format of each class of chapter entries.

synopsi ::= header citations synopsis_text [methods]

E&V_technology ::= prolog method

C.2.3 Formal Chapter Ordering

The formal portion of the GB is found in Chapters 4 through 16.

formal_chapters ::= [synopsis]*
[E&V_technology]*
.
.
.
[E&V_technology]*

C.3 TABLE OF CONTENTS

The table of contents shares some features with the rest of the formal aspects of the GB.

table_of_contents ::= [chapter]*

chapter ::= designator phrase designator_page

designator_page ::= designator "-" digits

C.4 CITATIONS

The citations are found in Appendix A, and have a formal structure as defined in the following grammar. The (semantic) form of citation text is taken from the standard for IEEE Software Magazine.

citations ::= [citation]*

citation ::= key body "."

key ::= "[" phrase_list "]"

body ::= [reference_list] phrase_list

APPENDIX D
VENDORS AND AGENTS

[ACM]
ACM Order Dept. (800) 342-6626
P.O. Box 64145 (301) 528-4261
Baltimore, MD 21264

[ACSI]
Advanced Computing Solutions, Inc. (713) 280-9917
17049 El Camino Real, Ste. 202
Houston, TX 77058

[Aerospace]
Richard Ham (213) 336-3438
Aerospace Corporation
P.O. Box 92957
Los Angeles, CA 90009

[AIM]
AIM Technology (800) 848-8649
4699 Old Ironsides Drive, Suite 150 (408) 748-8649
Santa Clara, CA 95054
EMail: UUCP benchinfo@aimt.uu.net

[AJPO]
The Ada Joint Program Office (703) 614-0208
Rm 3D139
(Fern St/C107)
The Pentagon
Washington, D.C. 20301-3081
EMail: fittsd@ajpo.sei.cmu.edu

[ARTEWG]
Mike Kamrad (612) 456-7315
Unisys Computer, Systems Division
MS U2F13
P.O. Box 64525
St. Paul, MN 55164-0525
EMail: mkamrad@ajpo.sei.cmu.edu

[AVF]
Mr. Bobby R. Evans (513) 255-4472
ASD/SCEL
Wright-Patterson AFB
OH 45433
EMail: evansbr@wpafb-jalcf.arpa

[Boeing Aerospace]
Thomas Bowen, Gary Wigle, Jay Tsai
Boeing Aerospace Company
P.O. Box 3999
Seattle, WA 98124

[BSI]
British Standards Institute 0908 220908
Information Technology Department
BSI Quality Assurance
P.O. Box 375 Linford Wood
Milton Keynes MK14 6LL
United Kingdom

[BYTE]
Mary Ann Goulding (603) 924-2664
BYTE Publications
One Phoenix Mill Lane
Peterborough, NH 03458

[Cambridge University Press]
Cambridge University Press
32 East 57th Street
New York, NY 10022

[CECOM]
Mr. Edward Gallagher (412) 268-5758
U.S. Army CECOM
AMSEL-RD-SE-AST
Ft. Monmouth, NJ 07703
EMail: egallagh@ajpo.sei.cmu.edu

[College of Wales]
Dr. Fred W. Long
Department of Computer Science
University College of Wales
Aberystwyth, UK

[ComputerWorld]

Margaret McIndoe (508) 879-0700
ComputerWorld
Back Issues
Box 9171, 375 Cochituate Road
Framingham, MA 01701-9171
EMail: MCI COMPUTERWORLD

[Draper]

James K. Van Buren, Jr. (617) 258-2722
MS 3C
The Charles Stark Draper Laboratory
555 Technology Square
Cambridge, MA 02140

[DTIC]

Defense Technical Information Center (202) 274-6871 (Registration)
Cameron Station (703) 274-7633 (Reference)
Alexandria, VA 22314

[DACS]

Data Analysis Center for Software (315) 336-0937
RADC/COED
Bldg 101
Griffiss AFB, NY 13441-5700
Attn: Document Ordering
EMail: dacs@radc-multics

[E&V Team]

Mr. Raymond Szymanski (513) 255-6548
WL/AAAF -3947
Wright-Patterson AFB AV 785-6548
OH 45433-6543 -3947
EMail: szymansk@ajpo.sei.cmu.edu

[GIT]

Georgia Institute of Technology (404) 894-3180
Software Engineering Research Center
Atlanta, GA 30332-0280

[IEEE Computer]

IEEE Service Center (201) 981-0060
445 Hoes Lane
P.O. Bos 1331
Piscataway, NJ 08855-1331

[InfoWorld]
InfoWorld (415) 328-4602
1060 Marsh Road, Suite C-200
Menlo Park, CA 94025

[INSTEP]
Mr. Brian A. Nejmeh (703) 742-6276
Innovative Software Engineering Practices, Inc.
13526 Copper Bed Road
Herndon, VA 22071
EMail: nejmeh%instep@uunet.uu.net

[MITRE VA]
MITRE Corporation (703) 883-6000
Civil Systems Division
7525 Colshire Drive
McLean, VA 22102-3481

[NADC]
Naval Air Development Center (215) 441-2000
Street & Jacksonville Roads
Warminster, PA 18974-5000

[NASA/Ames]
Information Sciences Division
MS 244-4
NASA Ames Research Center
Moffett Field, CA 94035

[NTIS]
National Technical Information Service (703) 487-4650
U.S. Department of Commerce (202) 724-3374
5285 Port Royal Road
Springfield, VA 22161

[PC Magazine]
PC Magazine (212) 503-5255
One Park Avenue
New York, NY 10016
EMail: MCI PC Magazine

[PC World]
PC World Communications, Inc. (415) 243-0500
501 Second Street
San Francisco, CA 94107
EMail: MCI PCWORLD

[Personal Computing]
Personal Computing Magazine, Inc. (800) 829-9097
999 Riverview Drive
Totowa, NJ 07512

[PIWG]
Mr. Rob Spray (214) 907-6640
PIWG Tree
P.O. Box 850236
Richardson, TX 75085-0236

[RADC]
Rome Air Development Center (COEE) (315) 330-4654
Griffiss AFB, NY 13441-5700

[SAIC]
Gregory T. Daich (619) 546-6000
Science Applications International Corporation
10260 Campus Point Drive
San Diego, CA 92121

[SERC]
Ronnie J. Martin (317) 494-6012
Software Engineering Research Center
Purdue University
Department of Computer Sciences
West Lafayette, IN 47907-2004
EMail: rjm@purdue.edu

[SEI]
Software Engineering Institute (412) 268-7700
Carnegie Mellon University
Pittsburgh, PA 15213

[SofTech OH]
Teresa L. Banks (513) 429-3241
SofTech, Inc.
3100 Presidential Drive
Fairborn, OH 45324-2039
ARPAnet: hillm@wpafb-jalcf

[SofTech TX]
David Remkes (713) 480-1994
SofTech, Inc.
1300 Hercules #105
Houston, TX 77058
EMail: chilsonl@ajpo.sei.cmu.edu

[SPC]
Software Productivity Consortium
SPC Building
2214 Rock Hill Road
Herndon, VA 22070

[SPS]
Software Productivity Solutions, Inc. (407) 984-3370
P.O. Box 361697
Melbourne, FL 32936

[STSC]
Rudy Alder (801) 777-7303
Manager, STSC AV 458-7703
OO-ALC/MMETI
Hill AFB, UT 84056

[TRW]
Lynn H. Satterthwaite
TRW
Military Electronics & Avionics
1104 Country Hills Drive
Ogden, UT 84403

[UMich]
Russel M. Clapp, Louis Duchesneau, (313) 764-1817
Richard A. Volz, Trevor N. Mudge, and Timothy Schultze
The Robotics Research Laboratory
The University of Michigan
Ann Arbor, MI 48109

REQUEST FOR USER FEEDBACK

Readers of this Guidebook and its companion, the "E&V Reference Manual," are urged to provide feedback by sending comments, either electronically or by surface mail. For electronic messages use:

szymansk@ajpo.sei.cmu.edu

For surface mail use:

Mr. Raymond Szymanski
WL/AAAF
Wright Patterson AFB, OH 45433-6543

This form provides a convenient mechanism for providing such feedback. Please, (1) copy the form as many times as needed, and use blank pages if you need larger spaces for your answers, (2) fill it in, and (3) mail it to the above address.

Thank you for your time and effort on behalf of E&V technology improvement.

USER IDENTIFICATION

Name (optional): _____
Phone (optional): _____
Address (optional): _____
_____
_____

Net Address (optional): _____

Please circle any that apply to your role or your employer:

Software Acquisition APSE/Tool User APSE/Tool Builder

E&V Technology User E&V Technology Builder Investor

Government Industry Academia Self-Employed

Why did you order the E&V Reference System documents? _____
_____
_____
_____

USE OF THE REFERENCE SYSTEM

Do you feel that the documents are useful?
E&V Reference Manual _____ E&V Guidebook _____

Have you already used them? _____ How? _____
_____
_____

Can your co-workers use them in other ways? _____ How? _____
_____
_____

GENERAL COMMENTS

What particularly useful items have you noticed? _____
_____

What additions would you like to suggest? _____
_____
_____

What are your general comments or criticisms? _____
_____
_____
_____
_____

COMMENTS ON SPECIFIC SECTIONS

Section Number: _____ (e.g., GB 6.4 for Guidebook Section 6.4)
Section Title: _____

Suggested Changes and/or Additions: _____

Section Number: _____ (e.g., GB 6.4 for Guidebook Section 6.4)
Section Title: _____

Suggested Changes and/or Additions: _____

Section Number: _____ (e.g., GB 6.4 for Guidebook Section 6.4)
Section Title: _____

Suggested Changes and/or Additions: _____